INTERNETWORK DOMAIN ACCESS CONTROL SECURITY

By

JYH-HAW YEH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1999

ACKNOWLEDGMENTS

I wish to express my whole-hearted appreciation and gratitude to my advisor, Professor Randy Chow, for giving countless hours of guidance in my research work. Without his support and patience, this research would not have been completed. Also special thanks to Dr. Richard Newman, for his valuable advice on my research during our weekly research group meeting.

I would also like to thank other members in my supervisory committee, Dr. Tim Davis, Dr. Eric Hanson, and Dr. Mark Yang, for their interest and comments.

Last, but not least, my greatest appreciation goes to my parents and my wife, Sylvia. Without their love, support, and encouragement, I would not have made it this far. I am always surrounded by their love and tender care. To them I dedicate this work.

This research was supported, in part, by the National Security Agency under grant MDA-904-98-C-A892.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abstract of Dissertation
Presented to the Graduate School of the University of Florida
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

INTERNETWORK DOMAIN ACCESS CONTROL SECURITY

By

Jyh-haw Yeh

December 1999

Chairman: Dr. Randy Chow
Major Department: Computer and Information Science and Engineering

An internetwork consists of heterogeneous domains managed under different administrative authorities. For secure interdomain resource sharing, it is necessary to implement an Interdomain Access Control (IAC) protocol to regulate traffic flow between end domains and transit domains. Control of traffic flow from one stub (end) domain to another must carry some proof of authentication and authorization to exit from, enter to, and transit through a domain. This authentication and authorization proof usually needs cryptographic encryption and decryption. Thus, safe session key distribution becomes a preliminary for an IAC protocol to succeed. A key assignment scheme for internetwork domains is proposed so that the safe session key distribution can be accomplished.

An end-to-end authentication and authorization protocol is not sufficient unless the policy and resource requirements at all transit domains are also considered. This issue is closely related to internetwork routing protocols. Therefore, when designing an interdomain access control protocol it is logical to integrate the protocol with underlying network routing facilities. Interdomain Policy Routing (IDPR) suggested by RFC 1479 [24] is a typical such routing facility for current packet switched networks. Two interdomain access control protocols, KIAC (Key-based IAC) and TIAC (Ticket-based IAC), are proposed. Both protocols can be built on top of IDPR.

In the implementation of IAC protocols with IDPR, some state information of a connection must be maintained and some computation must be performed by the routers at transit domains. This perfectly fits the emerging active network and mobile agent technologies with the store-compute-forward networking paradigm. A dynamic interdomain path setup for IAC protocols is also proposed under the active network infrastructure.

CHAPTER 1
INTRODUCTION

An internetwork is composed of many administrative domains (ADs). Network traffic flows between *stub* ADs through intermediate *transit* ADs. Since stub ADs may have competing interests, each end-to-end communication session between two hosts in different stub domains should be authenticated and authorized.

Traditional firewall schemes [4] achieve some degree of interdomain access control. In a firewall system, network traffic filtering is performed on a per-packet basis independently by firewalls in each domain. The authorization process for every packet in a firewall may be time-consuming due to complex domain policies and the variety of the requested types of service.

This research concerns establishing secure and authorized communication sessions between hosts with cooperating domain gateways (routers) along the communication paths. Not only do connection-oriented interdomain access control (IAC) protocols offer more flexibility and implementation efficiency for access control policies than do firewalls, they also lower the internetwork load by reducing the amount of allowable traffic.

End-to-end interdomain access control protocols are analogous to international travel. To travel from the United States to Germany via France, a traveler needs to obtain a US passport, an entry visa to Germany and possibly a transit visa for stopping over in France. Similarly, a packet flowing from one stub domain to another must carry some proof of authentication and authorization to exit from, enter to and transit through a domain. Several such protocols (e.g., VISA [7] and PASS [16]) have been proposed. Since these protocols are connection-oriented, they consist of an initialization and a packet forwarding phases. During the initialization phase, the protocols establish a shared session key between two stub domains. In the packet forwarding phase, each packet carries a message authentication code (MAC) [13] signed by the secret session key to prove its authenticity and authorization to the participating domain gateways. Therefore, a secure interdomain communication session strongly relies on how safely the session key can be distributed to the participating domain gateways through networks. In Chapter 2, a long term secret key shared between every two neighbor domain gateways is assumed so that the session key can be distributed among domain gateways in an encrypted form using the shared secret keys. This assumption is not attractive because some critical transit backbone domains may need to maintain a large number of secret keys if there are many neighbor domains connected to them. Therefore, a key assignment scheme is developed in which each node (domain) only needs to remember two keys. The key assignment scheme is described in Chapter 4, and Chapter 5 demonstrates how the scheme can be utilized in interdomain access control.

Interdomain access control protocols are intended to form a mandatory control framework for all network applications that require information flow across heterogeneous domains. Applications such as remote login, file transfer, and e-mail fall into this category and can be enhanced with additional security using an IAC protocol. The primary motivation behind an IAC mechanism is to control access to valuable network resources. In the context of interdomain communication, not only are the stub ADs' network resources valuable, but also the transit ADs'. Access control schemes such as VISA and PASS focus only on access control to the stub domains; transit domain gateways perform no active role in these protocols. Other protocols such as the Packet Level Access Control by Iqbal [12] take transit domains into consideration. However, this protocol does not address routing decisions in a dynamic routing environment. Routing decisions in transit domains are separated from the access control between stub domains.

To establish a feasible communication path between two stub domains, the local routing policy in each transit domain must be considered. Many policy routing protocols [24, 21, 19, 18] exist. We use the interdomain policy routing (IDPR) in our proposed IAC protocols in Chapter 2 to integrate the implementation of end-to-end access control with policy routing in transit domains, though any of these protocols would suffice.

The Path Setup and Packet Forwarding protocols in IDPR are closely related to interdomain access control. The Path Setup protocol is responsible for computing a communication path based on the routing policies contained in the routing information databases (RIDs), and distributing a Path ID to each of the involved domain (stub and transit) Policy Gateways (routers) along the communication path. In the Packet Forwarding protocol, every data packet is checked against the Path IDs stored in the forwarding information database (FID). Only packets with a valid Path ID are let through.

Path IDs are for routing purposes only. They cannot be used to compute MACs to protect against spoofing or illegal modification of packets. Our first secure interdomain access control protocol key-based IAC (KIAC), presented in Chapter 2, is a straightforward extension of IDPR with secret session keys in place of Path IDs. RIDs and FIDs are the two primary sources of storage overhead in the implementation of IDPR and KIAC. Because of these, interdomain routing and access control does not scale well. Many approaches have been proposed (superdomain in [24, 18], viewserver hierarchy in [2], and landmark hierarchy in [29]) to reduce the RID overhead. Our second secure interdomain access control protocol ticket-based IAC (TIAC), presented in Chapter 2, eliminates the FID storage overhead.

Both KIAC and TIAC are built on top of IDPR, which builds a communication path using a static path determination strategy. The static path determination strategy means that only the source router determines the path. In order to achieve this, each router should maintain a consistent RID that contains the connectivity of the internetwork and the associated domain policies of each node so that a feasible path can be computed. The large storage overhead of RIDs and the difficulty of maintaining the RIDs' consistency among routers make this approach scale poorly. Therefore, a dynamic interdomain communication path setup protocol using a dynamic path determination strategy is proposed in Chapter 3 to eliminate the necessity of RIDs. In contrast to the static approach, the dynamic approach shifts the path determination from one node to a set of nodes, or from centralized to distributed. Another dynamic feature of this protocol is that a path can be reconfigured to bypass a failed or congested router/link.

The dynamic functionality of the proposed protocol greatly increases the computational responsibility of the intermediate network nodes. This coincides perfectly with the network resolution trend from passive data network to active network [26, 25, 32, 3] in which the network is treated as a computing engine. In the IAC protocols, some state information about a connection must be maintained and some computation must be performed by the routers at transit domains. This leads to investigation of the emerging *active network* and *mobile agent* technologies for the implementation of IAC protocols.

The active network is a revolutionary concept that assumes a store-compute-forward networking paradigm. Packets traversing the Internet are allowed to carry both program and data. At each network node that a packet visits, the program is executed on the data and local state, possibly leaving some state information for subsequent packets, before the resulting packet(s) is forwarded to the next node. In effect, each router actively performs some network/application-related computation in addition to traditional routing. In this scheme, the programs are similar to mobile agents that are interpreted at each network node. The entire network becomes programmable. An immediate benefit is that new protocols (including IAC) can be deployed easily and co-exist with existing protocols. Many innovative applications such as Web caching and network multicasting also can be effectively supported by an active network. Therefore, we believe that active networks provide the most suitable network infrastructure to implement the proposed dynamic protocol for secure access control over the Internet.

CHAPTER 2
INTERDOMAIN ACCESS CONTROL WITH POLICY ROUTING

This chapter proposes two secure interdomain access control protocols that are integrated with an interdomain policy routing protocol. With the ability to compute a feasible communication path in advance, each domain policy gateway (PG) can build a logical connection along the path in the initialization phase. Thus, not only can the stub administrative domains (ADs) enforce their policies, so can the transit ADs. Traffic flow control over all domain PG in the path is desirable in a heterogeneous internetwork environment. This can prevent the unpredictable behavior of dropping packets by transit PGs. The packet dropping problem occurs on VISA, PASS, and any other protocol that only builds a logical connection for two stub ADs.

Key-based and ticket-based interdomain access control (KIAC, TIAC) protocols provide a uniform and more efficient way to achieve integrity, authenticity, and privacy than interdomain policy routing (IDPR). Both IDPR and KIAC suffer from heavy memory overhead in PGs. For some topologically critical transit PGs, this memory overhead will become huge and unmanageable. TIAC is designed to eliminate this memory overhead without significantly degrading the performance and security.

The chapter is organized as follows: Section 2.1 describes the basics of the IDPR Path Setup and Packet Forwarding protocols. Section 2.2 presents the proposed secure interdomain access control protocols, KIAC and TIAC. Section 2.3 compares the performance overhead of the two protocols in terms of network load, cryptographic computation and key searching complexities. A summary is given in Section 2.4.

2.1 InterDomain Policy Routing (IDPR)

A policy route (PR) or path in IDPR is a sequence of domain PGs. Under the IDPR architecture, each domain PG maintains a routing information database (RID) which contains transit policies (transit service offered) for all other domains in the internetwork. With this database in hand and type of service requested, each domain PG has the ability to compute a feasible PR for any interdomain traffic flow, if one exists. Based on a PR, the Path Setup Protocol establishes the actual communication path. The routing information is contained in the FID for use by the Packet Forwarding Protocol. We will briefly describe these two protocols in the following subsections.

2.1.1 Path Setup Protocol

Assume a generic internetwork topology as shown in Figure 2.1. When host H_a in domain D_1 wishes to communicate with host H_b in domain D_n, H_a sends out a data packet with destination address H_b. This data packet will be forwarded within D_1 by an intradomain routing protocol. Eventually, it will reach PG_1 in D_1. Note that PGs are the only entry or exit points for a domain.

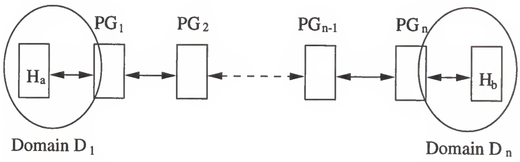


Figure 2.1. Host H_a communicates with Host H_b

PG_1 uses the data packet's header, including source and destination addresses, to determine whether it is an interdomain data packet. If it is indeed an interdomain data packet, PG_1 will first try to locate an entry in its own FID for this data packet. It uses some information (Path ID, source/destination addresses) in the data packet's header as search indices. This search will result in an entry for an existing traffic flow, or fail in case of a new traffic flow. If an entry is found, PG_1 forwards this data packet according to the entry. The format of the FID's entry is shown in Table 2.1. Path ID field contains a unique path identifier for a traffic flow. Src (Dest) Host field specifies the source (destination) host address. Prev (Next) PG field specifies the previous (next) PG in the PR. The last field, Next Hop, indicates the next router to which this traffic flow should be forwarded.

Table 2.1. Format of FID entry in IDPR

Path ID	Src Host	Dest Host	Prev PG	Next PG	Next Hop

If no entry is found in the FID, PG_1 generates a feasible PR by consulting transit domains policies in its RID. The computation of PR uses inputs such as source/destination domain identifiers, requested service, source domain policy, and transit domain policies. Based on this computed PR, PG_1 starts to set up the communication path. Without loss of generality, we assume the PR is PG_1–PG_2–$PG_3 \ldots PG_n$ (see Figure 2.1).

The path setup works as follows: PG_1 invents a new Path ID and updates its FID by adding an new entry for this Path. Then, PG_1 sends a Path Setup message to PG2 with information such as Path ID, PR, requested services, and source/destination addresses. Upon receiving this Path Setup message, PG_2 uses information in this message to check against its own transit policy to determine whether to accept or reject this path. In case of acceptance, PG_2 forwards this message to PG_3 and updates its FID as described earlier. This procedure continues until PG_n accepts this path and updates its FID. Finally, PG_n returns an ACCEPT packet all the way back to PG_1 in the reverse direction to notify all

PGs in the path. After PG_1 has received the ACCEPT packet from PG_n, each PG in this route should have an entry in its FID corresponding to this path. Packet forwarding is ready to proceed. Any failure during this path setup causes a REJECT packet to be sent back to PG_1. PG_1 should either compute another feasible PR if there exists one or send a REJECT packet back to the source host otherwise.

2.1.2 Packet Forwarding Protocol

Whether H_a should possess the Path ID or not is unspecified in the IDPR specification. To be consistent with the IAC protocol discussed later, we assume that H_a gets the Path ID from PG_1 during Path Setup. H_a attaches the Path ID to every data packet bound for H_b. PG_1 will try to locate a FID entry using the Path ID as an index. In case of success, PG_1 knows the next PG (in this example, PG_2) through which this data packet should travel. PG_1 encapsulates the received data packet so that PG_2 becomes the destination in the outer header and forwards this encapsulated packet to the next hop. All other PGs perform the same procedure as PG_1. Ultimately, this data packet reaches the PG_n and H_b in domain D_n.

2.1.3 IDPR Security

The IDPR protocol provides an INT/AUTH header field for integrity and authenticity purposes. Thus, all IDPR packets can be protected by this header field to prevent address spoofing and illegal modification. Note that the purpose of this field is the same as the message authentication code (MAC) we use. We will use the term MAC hereafter.

Since there exists no session key shared among all PGs in the path, the MAC of a data packet needs to be recomputed with the shared secret key between peer PGs. This computational overhead can be significant.

2.1.4 IDPR Summary

IDPR is a typical policy routing protocol that entails establishment of a communication path using a static path determination strategy. Static path determination means that only the source router determines the path. Each router must maintain a consistent RID, containing the connectivity of the internetwork and the associated domain policy of every node, for the computation of feasible paths. Note that assuming the consistency among all RIDs and correctness of Path Computation, the routing path computed by the source router for a communication session is feasible since it most likely will be agreed on by all transit routers. Having the ability to compute a feasible routing path in a source router, the path setup can confidently commence without the fear of rejection.

The Path Setup protocol is responsible for setting up a communication path based on the source computed PR, and distributing a Path ID to each of the involved domain PGs along the path. In the Packet Forwarding protocol, every data packet is checked against the Path IDs stored in the FID. Only packets with a valid Path ID are let through.

Path IDs are for routing purposes only. They cannot be used to compute MACs to protect against spoofing or illegal modification of packets. Therefore, secure IAC protocols are necessary to enhance the security functionality of IDPR. In the next section, two such kind of protocols, KIAC and TIAC, are proposed.

2.2 Secure Interdomain Access Control Protocols

In the IDPR protocol, each packet's authentication and integrity are achieved by a MAC recomputation at each domain gateway along the communication path. This performance overhead can be reduced if there is a shared session key instead of a path ID. In this section, two protocols KIAC and TIAC with different approaches of sharing session keys are addressed.

2.2.1 Assumptions and Notation

Before the discussion of the proposed KIAC and TIAC protocols in Sections 2.2.2 and 2.2.3, we define the following assumptions and notation.

2.2.1.1 Assumptions

The assumptions below are common to both KIAC and TIAC.

1. The proposed protocols are based on a symmetric key encryption algorithm to ensure authenticity, privacy, and integrity.

2. Every PG shares a secret key with each host within its domain.

3. Adjacent PGs also share a secret key.

4. Each PG has its own secret key.

5. All these secret keys listed above are already in place and stored securely.

Assumptions 2, 3, 4, 5 define which entity should share a secret key with another among all PGs and hosts in a PR so that both KIAC and TIAC are capable of distributing a session key using symmetric encryption algorithms.

2.2.1.2 Notation

We will illustrate our two protocols using the example in Figure 2.1. The notation used in this section is given below.

K_{a1}*:* secret key shared by H_a and PG_1

K_{12}*:* secret key shared by PG_1 and PG_2

...

$K_{(n-1)n}$*:* secret key shared by PG_{n-1} and PG_n

K_{nb}: secret key shared by PG_n and H_b

K_i: secret key known only to PG_i, for i=$1, 2, 3, \ldots, n$

K_s: secret session key for a traffic flow

T^i_{ba}: i-th partial ticket generated by PG_i for traffic from H_b to H_a, for i=$1, 2, 3, \ldots, n$. (If $i = n$ then it is a complete ticket.)

T^i_{ab}: (n-i+1)th partial ticket generated by PG_i for traffic from H_a to H_b, for i=$n, n-1, \ldots, 1$. (If $i = 1$ then it is a complete ticket.)

$MAC(K)$: packet's Message Authentication Code is a signed message digest of the packet's contents using key K. (K is one of the above peer shared secret keys or K_s)

$E(X, K)$: plaintext X encrypted by key K.

2.2.2 Key-based Interdomain Access Control Protocol (KIAC)

KIAC is a straightforward extension of IDPR with secret session keys in place of Path IDs for the security purpose. An initialization and a packet forwarding phases are corresponding to the path setup and the packet forwarding protocols in IDPR for setting up (initializing) a path and forwarding data packets.

These two phases are described in the following two sections.

2.2.2.1 Initialization Phase

Five control messages (with message contents enclosed in curly brackets) are used during the initialization phase. A REJECT message is the first control message used in the protocol, in response to an unauthorized data packet.

1. REJECT: $PG_1 \rightarrow H_a$;

{reason for rejection, MAC(K_{a1})}

2. AUTH-REQ: $H_a \rightarrow PG_1$;

$\{H_b$, requested service, $\text{MAC}(K_{a1})\}$

3. CONN-REQ$_{i(i+1)}$: $PG_i \rightarrow PG_{i+1}$;

$\{H_a$, H_b, PR, requested service, $\text{MAC}(K_{i(i+1)})\}$, for $i=1, 2, \ldots, n-1$.

4. CONN-CON$_{i(i-1)}$: $PG_i \rightarrow PG_{i-1}$;

$\{H_a$, H_b, $\text{E}(K_s, K_{(i-1)i})$, $\text{MAC}(K_{(i-1)i})\}$, for $i=n, n-1, \ldots, 2$.

5. KEY-FORWARD: $PG_n \rightarrow H_b$; $\{H_a$, $\text{E}(K_s, K_{nb})$, $\text{MAC}(K_{nb})\}$, or

$PG_1 \rightarrow H_a$; $\{H_b$, $\text{E}(K_s, K_{a1})$, $\text{MAC}(K_{a1})\}$

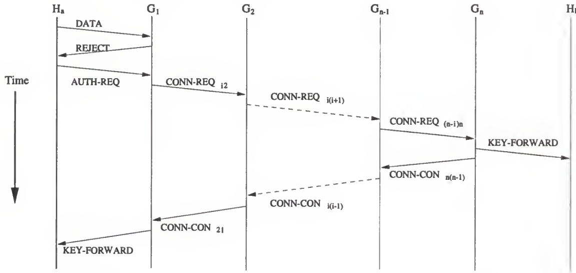


Figure 2.2. Illustration of the connection establishment for H_a and H_b

For H_a in D_1 to communicate with H_b in D_n, H_a sends a data packet bound for H_b (see Figure 2.2). Since this data packet does not yet have proper authorization (i.e., $\text{MAC}(K_s)$), PG_1 will reject it with a REJECT packet. Upon receiving the REJECT packet, H_a sends an authorization request AUTH-REQ packet to PG_1 indicating its request to access H_b. PG_1 starts the authorization process if the attached $\text{MAC}(K_{a1})$ in AUTH-REQ is correct. If this request is approved by the authorization process, PG_1 computes a feasible PR: (PG_1–PG_2–$PG_3 \ldots PG_n$) and adds an entry into its session key database (SKD) with the Src/Dest Host, Prev/Next PG, Next Hop, and Life Time fields. Each PG assigns a Life Time that represents the duration of validity of this communication session. The format of the SKD entry is shown in Table 2.2. (Note that if the control granularity is per communication session instead of per host pair, Path ID can be added as in IDPR.)

Table 2.2. Format of SKD entry in KIAC

Src Host	Dest Host	Prev PG	Next PG	Next Hop	Sess Key	Life Time

After the SKD entry has been added, PG_1 sets up a communication path by sending a connection request CONN-REQ$_{12}$ packet to PG_2 (the next PG in PR). CONN-REQ$_{12}$ packet contains a source/destination address pair, PR, and requested service so that PG_2 has enough information to proceed with authorization. Upon receiving the CONN-REQ$_{12}$, PG_2 verifies the MAC(K_{12}) of this request packet. In case of successful MAC verification, PG_2 initiates its local authorization process. The authorization process will return a negative answer only if PG_1's RID is inconsistent with PG_2's policies, since the computation of the PR has already taken all transit domains policies into consideration. This is the main reason why we integrate interdomain access control protocols with IDPR (or other interdomain routing protocols). If the request is authorized, PG_2 adds an SKD entry and fills in fields as PG_1 did, and continues path setup by sending another CONN-REQ$_{23}$ packet to PG_3 containing the same information as CONN-REQ$_{12}$.

The same procedure is repeated until PG_n receives the CONN-REQ$_{(n-1)n}$ request packet. As other PGs, PG_n verifies the MAC($K_{(n-1)n}$) first and makes a decision whether to accept the request. If the request is approved, PG_n invents a session key K_s, and adds an SKD entry with all fields. At this point, PG_n generates a connection confirmation

CONN-CON$_{n(n-1)}$ packet back to PG_{n-1} containing the source/destination host address pair, and encrypted session key. PG_{n-1} in turn generates another CONN-CON$_{(n-1)(n-2)}$ packet and sends it back to PG_{n-2}. These connection confirmation packets are propagated all the way back to PG_1. For each i, PG_i checks the authenticity and integrity of the connection confirmation packet by verifying the MAC and fills in the Session Key field by decrypting the session key encrypted with $K_{i(i+1)}$ and then reencrypting it with $K_{(i-1)i}$.

Both PG_1 and PG_n will generate a KEY-FORWARD packet to H_a and H_b, respectively. KEY-FORWARD packets contain the host address of the peer and the encrypted shared session key. Like PGs, H_a and H_b also maintain a SKD and add an entry when receiving the KEY-FORWARD packet.

2.2.2.2 Packet Forwarding Phase

After all entities have a SKD entry for this communication path, data packet flow commences. Since H_a has K_s in hand, it can send a data packet attached with MAC(K_s) without fear of rejection. Verifying the MAC(K_s) is enough to prove the data packet's integrity and authenticity (hence authorization). After verification the packet is forwarded to the next PG. Eventually, the data packet will reach its final destination H_b.

2.2.2.3 KIAC Security

The MAC of each control and data packet provides the authenticity and integrity checking to prevent address spoofing and illegal modification. The session key is encrypted using peer shared secret keys in the initialization phase and must be stored securely. Thus, a malicious eavesdropper can not obtain this session key to illegally consume valuable network resources.

Since a single session key is shared among all entities after initialization, the computation of the MAC for a data packet takes place only at the source host. It is not necessary for every PG to recompute a new MAC. From this point of view, KIAC is more efficient than IDPR.

2.2.3 Ticket-based Interdomain Access Control Protocol (TIAC)

Tickets used in this protocol are encrypted data containing the session key and other relevant information. Tickets are used to replace the SKD entries in KIAC thereby saving space at all intermediate PGs. Every interdomain data packet requires a valid ticket to pass through all PGs. In order to verify the attached MAC(K_s) of each data packet, a PG uses the peer shared secret key to decrypt the ticket and obtain the session key. Thus, the ticket must be carefully generated such that it can be decrypted by all PGs. Format of the partial/complete tickets is shown in Table 2.3. We need two complete tickets for a two way communication session, one for each direction. In our example, T_{ab}^n is for data packets from H_a to H_b, T_{ba}^n is for data packets from H_b to H_a.

Table 2.3. Format of (n-i+1)th and i-th partial Tickets

T_{ab}^i	Src Host H_a	Dest Host H_b	Next PG PG_{i+1}	Sess Key	Life-Time$_i$	Prev Partial Ticket $E(T_{ab}^{i+1}, K_{i(i+1)})$
T_{ba}^i	Src Host H_b	Dest Host H_a	Next PG PG_{i-1}	Sess Key	Life-Time$_i$	Prev Partial Ticket $E(T_{ba}^{i-1}, K_{(i-1)i})$

2.2.3.1 Initialization Phase

1. *REJECT:* $PG_1 \rightarrow H_a$;

   {reason for rejection, MAC(K_{a1})}

2. *AUTH-REQ:* $H_a \rightarrow PG_1$;

   H_b, {requested service, MAC(K_{a1})}

3. CONN-REQ$_{i(i+1)}$: $PG_i \rightarrow PG_{i+1}$;

$\{H_a$, H_b, PR, requested service, E($T^i_{ba}, K_{i(i+1)}$), MAC($K_{i(i+1)}$)}, for i=$1, 2, \ldots, n-1$.

4. CONN-CON$_{i(i-1)}$: $PG_i \rightarrow PG_{i-1}$;

$\{H_a$, H_b, PR, E($T^i_{ab}, K_{(i-1)i}$), MAC($K_{(i-1)i}$)}, for i=$n, n-1, \ldots, 2$.

5. KEY-FORWARD: $PG_n \rightarrow H_b$; $\{H_a$, E(T^n_{ba}, K_n), E(K_s, K_{nb}), MAC(K_{nb})}, or

$PG_1 \rightarrow H_a$; $\{H_b$, E(T^1_{ab}, K_1), E(K_s, K_{a1}), MAC(K_{a1})}

As in KIAC, H_a receives a REJECT packet from PG_1 because it sends out an interdomain data packet without a proper MAC(K_s). H_a sends an AUTH-REQ packet to PG_1 containing the address of destination host H_b and requested service. PG_1 checks the validity of MAC(K_{a1}) in AUTH-REQ and initiates its local authorization. If accepted, PG_1 computes a feasible PR. Meanwhile, PG_1 creates a session key and forms the 1st partial ticket T^1_{ba}. The Next PG and Previous Partial Ticket fields of T^1_{ba} should be empty or padded with zeros since they do not exist. PG_1 sends a CONN-REQ$_{12}$ packet to PG_2 containing the encrypted first partial ticket E(T^1_{ba}, K_{12}).

Each intermediate PG_i ($i = 2, 3, \ldots\ n-1$) executes the following sequence of steps after receiving a CONN-REQ$_{(i-1)i}$ packet:

1) Verify the MAC($K_{(i-1)i}$).

2) Authorize the request.

3) Decrypt E($T^{i-1}_{ba}, K_{(i-1)i}$) to obtain K_s.

4) Form the i-th partial ticket T^i_{ba} and encrypt it as E($T^i_{ba}, K_{i(i+1)}$).

5) Send CONN-REQ$_{i(i+1)}$ to PG_{i+1} with the same information as CONN-REQ$_{(i-1)i}$, but with newer encrypted i-th partial ticket.

Finally, PG_n also executes the same sequence of steps from 1 to 3, and forms the complete ticket T^n_{ba} and the opposite direction 1st partial ticket T^n_{ab}. PG_n sends the CONN-CON$_{n(n-1)}$ packet back to PG_{n-1} containing the encrypted 1st partial ticket E($T^n_{ab}, K_{(n-1)n}$) along with other information. The connection confirmation packets are propagated back to PG_1 and the partial ticket will grow into a complete ticket at PG_1. As in KIAC, both PG_1 and PG_n will send the KEY-FORWARD packet to H_a and H_b to distribute the complete ticket. Since the complete tickets are encrypted by secret keys only known to the end PGs, the end hosts have no way to know the session key from the tickets. Thus, the session key also should be distributed to end hosts in KEY-FORWARD packets. Note that only two end hosts H_a and H_b need to maintain the SKD in this protocol. Format of the SKD entry at the end hosts is shown in Table 2.4.

Table 2.4. Format of SKD entry in TIAC

Dest Host	Session Key	Encrypted Ticket

2.2.3.2 Packet Forwarding Phase

H_a starts the Packet Forwarding Phase by sending data packets to H_b with an encrypted ticket and MAC(K_s). Each PG can decrypt the ticket to obtain a session key, and use the session key to verify the MAC(K_s) to prove the data packet's integrity and authenticity (hence authorization). Next PG field in the ticket indicates the next destination address for encapsulation. The Life-Time field in the ticket ensures that this ticket has not expired. All fields, except the last one (Previous Partial Ticket), are stripped off by each PG. Since the ticket will be shrunk en route, the computation and verification of MAC(K_s) for the data packet must not include the ticket.

2.2.3.3 TIAC Security

An encrypted ticket is an identity-based capability. Thus, it is useful only for the source host. Each packet's MAC protects against address spoofing and illegal modification. Note that the computation of the MAC does not include the ticket. Therefore, illegal modification of this ticket cannot be detected by verifying the MAC. Fortunately, the encrypted ticket is tamper-resistant because it contains the Source Host and Destination Host fields. Every PG can decrypt the ticket and compare the source/destination addresses in the packet header. Any modification of the ticket can be detected if the match fails. Destination Host field in the ticket also can be used to prevent malicious ticket usage by the source host to communicate with other hosts within the same destination domain.

2.3 Performance Overhead

The analysis of performance overhead for KIAC and TIAC uses the following quantitative measurements:

1. Network load (NW): number of packets × number of PGs traversed
2. Computational complexity (COM): number of encryptions and decryptions
3. Secret key searching (SEC): number of secret key database searches
4. Session key searching (SES): number of session key database searches

The following assumptions are also used in the analysis:

1. The number of data packets for a communication session is M.
2. There are N policy gateways in the PR.
3. No REJECT packet is generated for this communication session.
4. The assumptions listed in section 2.2.1.

Tables 2.5 and 2.6 summarize the overhead of the initialization and packet forwarding phases, respectively.

The results indicate that TIAC requires more computation than KIAC, but less key searching. Which protocol is more efficient depends on various factors such as size of database, encryption algorithm used, and underlying machine architecture. For simplicity, if we assume that computation is comparable to key searching, then the overall performance

Table 2.5. Initialization phase overhead comparison

	NW	COM	SEC	SES
KIAC	$2N+1$	$6N+4$	$4N+1$	None
TIAC	$2N+1$	$8N+4$	$4N+1$	None

Table 2.6. Packet Forwarding phase overhead comparison

	NW	COM	SEC	SES
KIAC	MN	$M(N+2)$	None	$M(N+2)$
TIAC	MN	$M(2N+2)$	None	$2M$

overhead for TIAC is slightly higher than KIAC. However, the memory overhead of TIAC is greatly reduced because the required information is shifted from local memory to the ticket. Another performance concern for TIAC is the size of data packets, which contain an extra ticket beyond the MAC included in KIAC data packets. The size of a ticket is approximately 200N bits (25N bytes) if there are N layers in the ticket. For a typically 1000 to 1500 bytes MTU (Maximum Transfer Unit) link, the ticket could comsume $\frac{N}{40}$ to $\frac{N}{60}$ bandwidth in the link for a data packet. In TIAC, the number of layers in the ticket is equal to the number of gateways in the path because the ticket is encrypted layer by layer using the shared secret keys between every two neighbor gateways. Therefore, the ticket in a data packet could consume a major portion of the link bandwidth if the path is long. In Chapter 4, we develop a derivable key generation scheme that can be used to assign keys to Internet domain gateways to replace the shared secret keys between every two neighbor domain gateways. Also in Chapter 5, we demonstrate how to use the assigned keys to form the tickets such that the number of layers is at most three.

2.4 Chapter Summary

This chapter proposed two secure interdomain access control protocols that are integrated with an interdomain routing protocol. With the ability to compute a feasible

PR in advance, a PG can build a logical connection along the PR in the initialization phase. Thus, not only can the stub ADs enforce their policies, so can the transit ADs. Traffic flow control over all PGs in the route is desirable in a heterogeneous internetwork environment. This can prevent the undesirable behavior of dropping packets by transit PGs unpredictably. The packet dropping problem occurs on VISA, PASS, and any other protocol that only builds a logical connection for two stub ADs.

KIAC and TIAC protocols provide a uniform and more efficient way to achieve integrity, authenticity, and privacy than IDPR. Both IDPR and KIAC suffer from heavy memory overhead in PGs. For some topologically critical transit PGs, this memory overhead will become huge and unmanageable. TIAC is designed to eliminate this memory overhead without significantly degrading performance and security.

Unfortunately, a new network protocol generally requires a lengthy standardization process before it can be deployed in current network infrastructure. Furthermore, the RID memory overhead mentioned earlier also restricts their practical use in internetworks. In Chapter 3, a dynamic interdomain access control protocol in active networks is proposed to eliminate the necessity of a replicated RID and its deployment can be immediate.

CHAPTER 3
DYNAMIC INTERDOMAIN PATH SETUP IN ACTIVE NETWORKS

An internetwork is composed of many administrative domains (ADs) with different administrative and security policies for protecting their own valuable resources. A network traffic flow between stub ADs through intermediate transit ADs must not violate any stub or transit domain policy. Packets may be dropped by routers that detect a policy violation. Therefore, it is necessary for a communication session to set up a communication path in which all constituent routers are willing to serve for the session so that data packets can be delivered safely without being discarded.

Moreover, such a communication path cannot guarantee successful packet deliveries if the intermediate routers/links are subject to failures or congestion. A dynamic interdomain communication path setup protocol is proposed in this chapter to address these issues. The protocol is dynamic in the sense that the path determination strategy is distributed and a path can be reconfigured to bypass a failed or congested router/link. These two dynamic features require the intermediate network nodes to make some decisions and computation. The implementation of the protocol relies on the computational capability of active networks by which active nodes in the networks can provide computation in addition to traditional communication. Thus, the design of the protocol is based on the assumption of the active network architecture. The protocol will be a useful tool for all connection-oriented applications in active networks.

In this chapter, Section 3.1 describes the interdomain policy routing (IDPR) static path setup protocol. Section 3.2 briefly discusses the architecture of active networks. Section 3.3 presents the dynamic interdomain path setup protocol in active networks. A summary is given in Section 3.4.

3.1 Static Path Setup - Policy Routing

IDPR is a routing protocol designed for connection-oriented interdomain applications. It is composed of three primary protocols.

1. *Policy Update Protocol:* PUP handles reliable flooding of link state updates throughout the internetwork.

2. *Path Setup Protocol:* PSP installs and maintains routing information in all participating routers.

3. *Packet Forwarding Protocol:* PFP forwards data packets along a previously established path.

In IDPR, each router has a routing information database (RID) for storing the network topology of ADs and their associated transit policies. To provide a consistent view of the internetwork among all routers, the Policy Update Protocol maintains the consistency among RIDs by using a reliable flooding of link state updates throughout the internetwork. Another important component of IDPR, Path Computation, computes the routing path in accordance with source and transit domain policies. It is not a protocol in that each domain can implement its own version of Path Computation as long as it provides the interfaces to other protocols. Note that assuming the consistency among all RIDs and correctness of Path Computation, the routing path computed by the source router for a communication session is feasible since it most likely will be agreed on by all transit routers. Thus, an interdomain communication protocol can be implemented using the underlying IDPR policy routing facilities to determine a feasible routing path. Having the ability to compute a feasible routing path in a source router, the path setup can confidently commence without the fear of rejection.

In the Path Setup protocol, the source router sends out a Setup packet containing the computed path. Each transit router receiving this Setup packet checks its local administrative transit policy to determine whether to accept or reject this path. In case of acceptance, the router creates an entry for this session in its forwarding information database (FID). The purpose of this FID is for packet forwarding. An entry in FID consists of a path ID, previous and next router in the path, and other useful information. After an entry in FID is built, the router forwards the Setup packet to the next router in the path, and the process is repeated for the subsequent routers.

Once a path has been setup by the path setup protocol, the packet forwarding protocol forwards the user data packets through the path. Each router in the path uses the path ID and packet's MAC to check the authenticity and integrity of received data packets. For data packets with correct verification, the router forwards them to the next router recorded in the FID. In this way, all data packets for a session can flow through the established path to the destination.

IDPR Path Setup is static because the path is determined ahead of time statically and solely by the source. Each transit router can only accept or reject the proposed path. It plays no role in the path computation. This violates the general philosophy that the one providing the service should make the decision. Therefore, a dynamic path setup protocol is proposed in Section 3.3. In contrast to IDPR, each router in the proposed dynamic protocol determines the next segment (router) of the path.

3.2 Active Network Architecture

Active networks [26, 25, 32, 3] are a novel approach to network architecture in which each switch in the network provides a computation environment such that customized computation can be performed on the fly based on the messages flowing through them. In essence, the network becomes programmable for any specific application. Currently, a new network protocol generally requires a lengthy standardization process before it can be deployed. Under an active network architecture, the deployment of a new network protocol can be immediate.

Traditional data networks passively transport packets from one end system to another. The network does not care much about the contents of the packets it carries, and they are transferred between end systems without modification. Computation within the network is limited, e,g., header processing in packet-switched networks. The exponential growth of the Internet has brought diverse applications that may require intermediate network nodes to perform some computation on application data. For example, web browsing can be enhanced if the intermediate nodes support web page caching, and a path setup protocol needs to encrypt, decrypt, and validate packets at the intermediate nodes for safe key distribution.

These two unique features of the active network technologies, that routers are programmable for customized computation and new protocols are easily deployed, suit nicely the development and implementation of our proposed dynamic path setup protocol. Interdomain path setup not only is an application of active networks, it is also an essential tool for all connection-oriented applications in active networks. There is a strong synergy between the two.

The active network research group at MIT has identified two approaches to an active network architecture, discrete and integrated, depending on whether programs and user data are transported separately or in an integrated fashion [26, 25]. The proposed dynamic path setup protocol follows the integrated approach. To program a network, the integrated approach changes the passive packets of traditional network architectures to active capsules, which are programs with user data embedded. Capsules are active because they are executed at each router they traverse.

For the deployment of active networks, the design of active nodes are extremely important. Since active nodes execute foreign codes, they need a secure resource management mechanism to protect themselves from possibly malicious codes, as well as to enforce their own local resource policies. The resource management mechanism selectively allocates a certain amount of resources to each active capsule for a certain period of time. If any violation of active capsule execution occurs, the management mechanism should force the active capsule to terminate.

There are many issues and details in the design of active networks that are beyond the scope of this chapter. For its application in communication path setup, we will concentrate only on the programming with capsules for protocol implementation. To implement a path setup protocol in an active network, each router should be equipped as an active node. Since the underlying active network is a distributed computing engine, the determination of a feasible routing path can be decentralized. That is, the computation of a feasible routing path can be shifted from one node to a set of nodes, and from static to dynamic. Thus, each router no longer needs to maintain a large routing information database to keep track of the internetwork topology as in the Policy Routing approach.

The encoding of capsules has yet to be standardized by the active network research community. A functional description of capsules in our dynamic path setup protocol is given rather than detailed program codes.

3.3 Dynamic Path Setup

The proposed dynamic path setup protocol in active networks uses active capsules that contain control information for iterative negotiation of the next routers from source to destination through some qualified intermediate nodes. Once a path, which must not contain any routing loop, has been set up, the protocol is also responsible for the liveness of the connection by providing a path repair mechanism for reconfiguration of the path upon failures of link or router. Before the detailed description of the protocol in Sections 3.3.2 and 3.3.3, some data structures and packet types used in this protocol are described in the following subsection.

3.3.1 Soft State and Packet Types

In active network terminology, a "soft state" for a communication session specifies the current status of the session, and is maintained in each participating node. For the application in dynamic path setup, the soft state consists of four data structures in each node of the path. These data structures, listed below, either specify some useful information or record the status of the session.

Security Association (SA): This association includes the session ID, session key, encryption/decryption algorithms and all other information concerning security.

Qualified Neighbor List (QNL): A QNL in a node contains the available neighbors that are willing to provide the services for the session.

Nodes Traversed (NT): An NT is a sequence of nodes which have already been traversed by the Setup Capsule (the Setup Capsule is described later). During path setup, the Setup Capsule carries an NT, which is updated in each node. The usage of NT in the path setup is to avoid a routing loop. After the path has been built, NT contains the path and each node should have a copy of it.

Path Status (PS): PS is a two bit register that keeps track of the up/down status of the previous and next routers in the path.

Note that these four data structures in each node can be uniquely identified by the session ID.

Packets in this protocol are divided into two categories - capsule and message. A capsule carries a program for which a receiving router will fork a dedicated process to execute the program. A message only contains control information that is usually expected by a waiting process. There are four different capsules and eight different messages used in this protocol. All capsules and messages should carry a session ID for routers to identify the session.

Setup Capsule: A Setup Capsule is generated from the source host. It tries to build a routing path to the destination. A Setup Capsule contains the SRS (source requested service) and a data structure NT.

Repair Capsule: A Repair Capsule is generated if a failed or congested router/link is detected. This capsule is intended to find a detour route to bypass the failed or congested router/link. The Repair Capsule contains the SRS and two segments of the original NT separated by the failed or congested router/link.

Auth-Req Capsule: An Auth-Req Capsule is generated by the Setup Capsule or Repair Capsule in each router and broadcasted to all neighbor nodes to collect the authorization information about neighbors. An Auth-Req Capsule should contain the SRS.

Data Capsule: A Data Capsule carries the application data and a program for processing the data.

Yes/No Message: Upon receiving the Auth-Req Capsule, each neighbor router checks its local policy. A Yes/No Message is sent back indicating whether the policy allows the SRS. Based on these Yes/No Messages from neighbors, a QNL is built and the router can choose the next router to forward the Setup Capsule or Repair Capsule.

Grant Message: A Grant Message is generated from the destination router if a path is found. It is sent all the way back to the source through the path just found. The Grant Message carries the final NT (path) and the security association. Upon receiving the

Grant Message, each router stores the NT and the security association in local storage for future use.

Negative Message: A Negative Message is generated and sent back to the previous router in NT if a router could not find any feasible neighbor to forward the Setup Capsule or Repair Capsule. This occurs either when there is no qualified neighbor or all qualified neighbors send back Negative Messages.

Alive Message: After a path has been established, each router periodically sends an Alive Message to its two adjacent neighbors in the path to assert its viability.

Error Message: An Error Message is generated if a Data Capsule violates the authenticity and integrity checks based on the security association.

Repair Done Message: A Repair Done Message is generated if a detour path is found. It is sent back to the failure detecting router through the detour path. This message carries the new NT (path) and security association. Each router keeps the new NT and security association in local storage for future use.

Tear Down Message: A Tear Down Message requests the routers to release the memory allocated for the session. If the path can not be repaired, all routers should receive the Tear Down Message. On the other hand, if the path is repaired, all routers not in the new path should also receive the Tear Down Message.

NT Update Message: This message is to update the NT stored in the routers after a detour path has been built.

Setup Capsules, Auth-Req Capsules, Yes/No Messages, Grant Messages, and Negative Messages are used in a path setup. After a successful path setup, Data Capsules are used for application data forwarding, Alive and Error Messages are for failure detection, Repair

Capsules and Repair Done Messages are for detour route setup, and finally NT Update and Tear Down Messages are for state information consistency. The following two sections will discuss the usages of all the capsules and messages above in detail.

3.3.2 Path Setup

The network is treated as a computing engine in the active network architecture. For any network application, this computing engine needs some input programs from the source host and generates outputs for the application. In order to build a communication path, the program should instruct the engine to find a routing path to the destination in which all participating routers are willing to provide the requested service.

In the proposed path setup protocol, the input program is a Setup Capsule. The source host prepares and sends the Setup Capsule to the network engine. The Setup Capsule contains an empty NT at the beginning, and adds one router each time it traverses a router. When a router receives a Setup Capsule, it proceeds with the following procedures.

1. *Routing Loop Prevention:* The router checks the NT to see whether there is a routing loop. A routing loop exists if its own ID is already in the NT. In such a case, a Negative Message is sent back to the previous router and this process is terminated.

2. *Destination Router Process:* The router checks whether the destination host resides in its subnet. If yes, the router generates the security association and sends it along with the NT to the destination host and to the previous router in a Grant Message.

3. *Neighbor Information Collection:* The router broadcasts an Auth-Req Capsule containing the source requested service to all neighbor routers and waits for "Yes/No" responses. A QNL is built for all neighbor routers responding with a "Yes". Each neighbor router executes the Auth-Req Capsule by comparing its local policy and

the source requested service. A "Yes" Message is sent back if the policy allows the requested service. Otherwise, a "No" Message is sent back.

4. Next Router Selection Process: If the QNL is not empty, this procedure adds the current router to NT. It selects one neighbor router from the QNL until it is empty. For each selected neighbor router, two steps are performed.

(1) Forward the Setup Capsule to the selected neighbor router.

(2) Put the Setup Capsule Process into sleep and wait for Negative/Grant Messages. If a Negative Message is received, select another neighbor router and go to step (1). If a Grant Message is received, save the security association and NT in local storage and forward the Grant Message to the previous router or to the source host if the current router is the source router.

If all neighbor routers in QNL are selected and no Grant Message is received, a Negative Message is sent back to the previous router, the current router is deleted from NT, and this process is terminated.

The Setup Capsule generates the Auth-Req Capsule and broadcasts it to all neighbor routers. Without a careful design, this broadcasting technique could easily flood the Internet. The upper bound on the number of simultaneous active capsules for this one path setup is the number of neighbor routers. The path determination strategy in this protocol is dynamic because each router decides the next segment (router) of the path. The scenario for this strategy is depicted in Figure 3.1. For a clearer overall view of how this protocol works, the execution flow chart for Setup Capsule in each router is shown in Figure 3.2.

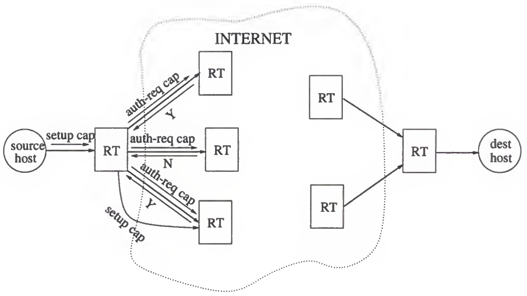


Figure 3.1. The scenario for dynamic path determination

3.3.3 Path Repair

As described earlier, another dynamic feature of the protocol is to bypass a failed or congested router/link during data transmission. In order to achieve this functionality, another input program for instructing routers to repair the path must be installed in each router before transmitting the user data. This path repair program can be carried in the Setup Capsule or another dedicated capsule. It is activated in each router when the router receives a Grant Message, i.e., the path is set. The path repair program basically has three procedures.

1. *User Data Forwarding:* After a path has been built, the path repair program expects Data Capsules from the source host. It checks each capsule's authenticity and integrity based on the security association. If the check is successful, the data processing program in the Data Capsule is called and executed. The path repair program resumes the control and forwards the Data Capsule after the called program is completed.

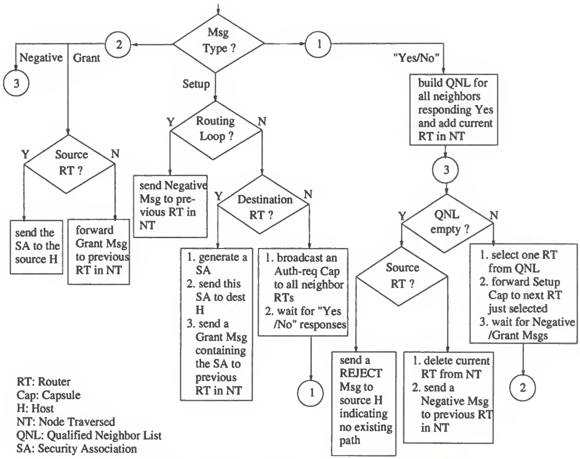

Msg Type ?
2
1
Negative
Grant
"Yes/No"
3
Setup
build QNL for all neighbors responding Yes and add current RT in NT
Routing Loop ?
Y
N
Source RT ?
3
send Negative Msg to previous RT in NT
Destination RT ?
QNL empty ?
send the SA to the source H
forward Grant Msg to previous RT in NT
1. generate a SA 2. send this SA to dest H 3. send a Grant Msg containing the SA to previous RT in NT
1. broadcast an Auth-req Cap to all neighbor RTs 2. wait for "Yes /No" responses
Source RT ?
1. select one RT from QNL 2. forward Setup Cap to next RT just selected 3. wait for Negative /Grant Msgs
send a REJECT Msg to source H indicating no existing path
1. delete current RT from NT 2. send a Negative Msg to previous RT in NT
RT: Router
Cap: Capsule
H: Host
NT: Node Traversed
QNL: Qualified Neighbor List
SA: Security Association

Figure 3.2. The execution of Setup Capsule in each router

2. *Failed Router/Link Detection:* Most of the time, the path repair program only performs the "still alive function" by sending and receiving Alive Messages. It periodically sends an Alive Message each to the previous and next routers in the path. A bit in the two bit register PS is set if the corresponding router's Alive Message is received within a default time threshold T. If both bits are set, PS is reset at the end of T. By examining the PS, a router can keep track of the Up/Down status of its adjacent routers in the path.

   Another potential failure is detected upon receiving an Error Message from the next router in the path after forwarding a Data Capsule to it. Consider the situation when the next router was down and came up again within the threshold T and the Up/Down protocol did not detect the failure. The soft state of this session would have been lost in next router and it could not recognize the forwarded Data Capsule. An Error Message would be returned by the next router.

3. Path Repair: If a failed router/link is detected by the Up/Down protocol, the router detecting the failure will select another neighbor in QNL and send a Repair Capsule to it. The scenario for issuing Repair Capsule in this case is shown in Figure 3.3.

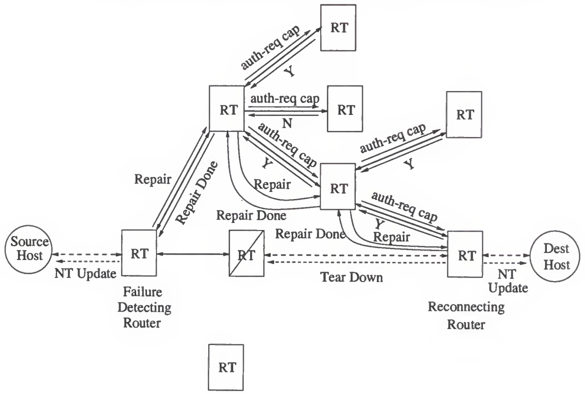


Figure 3.3. The scenario for a successful path repair

The Repair Capsule is the same as Setup Capsule except it finds a detour path from the failure detecting router to a reconnecting router. A reconnecting router can be any router posterior to the failed router in the original path. If the failed router/link is detected by receiving an Error Message, the Repair Capsule is sent to the next router to reconnect the path. In both cases, the Repair Capsule carries two segments of the original NT. The first segment of NT starts from the source router to the failure detecting router. This segment of NT is treated the same way as the NT in Setup Capsule, the router ID is inserted to the list when the capsule travels through a router. The second segment of NT contains the remaining routers of the original NT with the failed router marked. It is used to determine whether a new path has been found. A new NT can be computed from these two segments of NT when the path repair is completed.

When a router receives a Repair Capsule, it compares its ID to the two segments of NT. There are four possible results of the comparison.

(1) If its ID is the marked router in the original NT, the router simply rebuilds the soft state to reconnect the path for the session and sends a Repair Done message back to the failure detecting router.

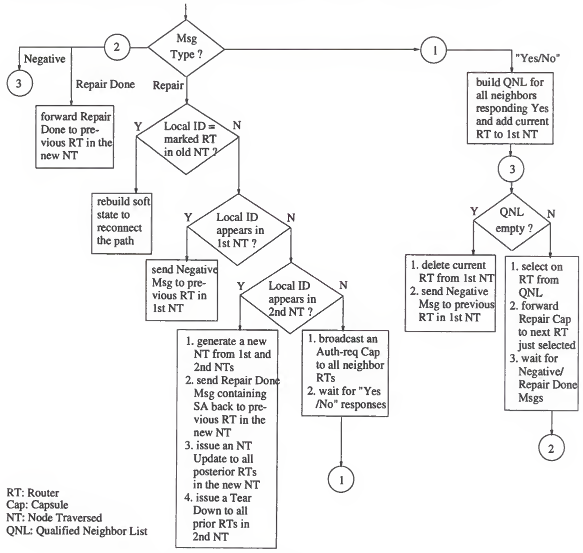
(2) If its ID appears in the first segment of NT, a routing loop exists and a Negative Message is sent back to the previous router in the first segment of NT.

(3) If its ID appears in the second segment of NT, the router is a reconnecting router and a detour path has been found. The sequence of routers posterior to the reconnecting router in the second segment of NT is appended to the first segment of NT to form a new NT for the new path. Then, a Repair Done Message containing the new NT is sent back to the failure detecting router through the new path. All routers in the new path need to update their state information by receiving either the Repair Done Message or NT Update Message. Also, all routers in the old path but not in the new path need to be released by receiving the Tear Down Message. More detailed usages of NT Update and Tear Down Messages will be discussed later in this section.

(4) If its ID does not appear in either of the two segments of NT, the same procedure for the Setup Capsule is applied to the Repair Capsule. That is, the router broadcasts

the Auth-Req Capsule, builds the QNL, selects a router from QNL, and forwards the Repair Capsule to the selected router.

The execution flow chart for Repair Capsule in a router is shown in Figure 3.4.



Msg Type ?
2
1
Negative
3
Repair Done
Repair
"Yes/No"
forward Repair Done to previous RT in the new NT
Local ID = marked RT in old NT ?
Y
N
build QNL for all neighbors responding Yes and add current RT to 1st NT
3
rebuild soft state to reconnect the path
Local ID appears in 1st NT ?
QNL empty ?
send Negative Msg to previous RT in 1st NT
Local ID appears in 2nd NT ?
1. delete current RT from 1st NT
2. send Negative Msg to previous RT in 1st NT
1. select on RT from QNL
2. forward Repair Cap to next RT just selected
3. wait for Negative/ Repair Done Msgs
1. generate a new NT from 1st and 2nd NTs
2. send Repair Done Msg containing SA back to previous RT in the new NT
3. issue an NT Update to all posterior RTs in the new NT
4. issue a Tear Down to all prior RTs in 2nd NT
1. broadcast an Auth-req Cap to all neighbor RTs
2. wait for "Yes /No" responses
2
1
RT: Router
Cap: Capsule
NT: Node Traversed
QNL: Qualified Neighbor List

Figure 3.4. The execution of Repair Capsule in each router

For a successful path repair, the consistency of the soft state NTs among all routers in the new path should be maintained as follows.

(1) Upon receiving the Repair Done Message, the failure detecting router issues an NT Update Message containing the new NT to all prior routers in the new path.

(2) After sending a Repair Done Message, the reconnecting router should issue an NT Update Message containing the new NT to all posterior routers in the new path and a Tear Down Message to all prior routers in the second segment of NT.

In the Up/Down protocol, two routers may detect a failed router/link simultaneously. Only the one closer the source issues the Repair Capsule. The one closer the destination should expect a Repair Capsule or a Tear Down Message for a default time threshold T'. If nothing is received within T', it means that the path repair is not successful and a Tear Down Message is issued to the routers posterior to it in the path.

If the failure detecting router receives a Negative Message, it means that the path repair is not successful. The router should try to send another Repair Capsule to another neighbor router in its QNL until the QNL is empty. If all neighbor routers are tried and not one is successful, the correct action is to either send a Tear Down Message all the way back to the source or send a Negative Message back to the previous router in the NT. The first choice stops searching another path and informs the source that there is no existing path at this moment. The second choice continues to search another detour path starting from the previous router in the NT. Which choice to select should depend on the upper layer applications.

The purpose of Tear Down Messages is to release the resources in the routers which are no longer serving for the session. In case of these routers do not receive the Tear Down Message because of another failure, the resource management mechanism in each active node should handle this after the life time of the session being expired. The execution flow chart for the path repair program in a router is shown in Figure 3.5.

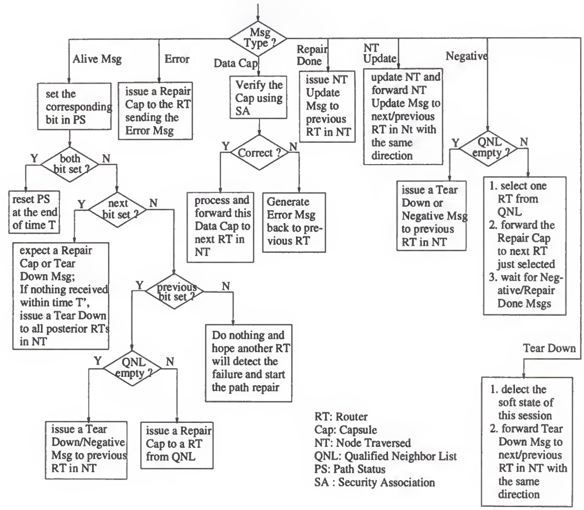

Msg Type ?
Alive Msg
Error
Data Cap
Repair Done
NT Update
Negative
set the corresponding bit in PS
issue a Repair Cap to the RT sending the Error Msg
Verify the Cap using SA
issue NT Update Msg to previous RT in NT
update NT and forward NT Update Msg to next/previous RT in Nt with the same direction
Y
both bit set ?
N
Y
Correct ?
N
Y
QNL empty ?
N
reset PS at the end of time T
Y
next bit set ?
N
process and forward this Data Cap to next RT in NT
Generate Error Msg back to previous RT
issue a Tear Down or Negative Msg to previous RT in NT
1. select one RT from QNL
2. forward the Repair Cap to next RT just selected
3. wait for Negative/Repair Done Msgs
expect a Repair Cap or Tear Down Msg; If nothing received within time T', issue a Tear Down to all posterior RTs in NT
Y
previous bit set ?
N
Do nothing and hope another RT will detect the failure and start the path repair
Tear Down
Y
QNL empty ?
N
1. delect the soft state of this session
2. forward Tear Down Msg to next/previous RT in NT with the same direction
issue a Tear Down/Negative Msg to previous RT in NT
issue a Repair Cap to a RT from QNL
RT: Router
Cap: Capsule
NT: Node Traversed
QNL: Qualified Neighbor List
PS: Path Status
SA : Security Association

Figure 3.5. The execution of path repair program in each router

In this protocol, there are three major programs running in each participating routers. These programs are the Setup Capsule, path repair program, and Repair Capsule. Table 3.1 briefly summarizes the differences among them.

3.4 Chapter Summary

A dynamic interdomain path setup protocol is presented in this Chapter. We assume that the underlying network architecture is the active network, a novel network architecture in which each intermediate network node is able to perform some customized computation. The protocol is different from others in that it utilizes a distributed path determination

Table 3.1. Comparison among Setup Capsule, path repair program, and Repair Capsule

	objective	relationship	running routers
Setup Capsule	set up a path	issued by source at the beginning	all routers receive this capsule
path repair program	detect the failure and maintain the path	activated after Setup Capsule is finished	all routers in the path
Repair Capsule	set up a detour route to bypass the failure	issued by the path repair pgm when detecting a failure	all routers receive this capsule

strategy and has an automatic path repair. The distributed path determination strategy shifts the decision from the source router to all intermediate routers.

We believe that the philosophy behind the strategy fits more precisely in the context of interdomain communication, i.e., the one providing the service makes the decision. Furthermore, the automatic path repair mechanism can detect failures much quicker since they are always discovered first by the nearest router. The nearest router can initiate the path repair process immediately upon detecting a failure. An active network environment facilitates both the computational and communication aspects of the protocol. Many active network applications will rely on such a connection setup protocol to establish active nodes for their respective computation.

Whether and how to reconfigure a communication path upon failures is an important protocol design issue. To repair a failed communication session, a fast path repair process is crucial. The process should be efficient in failure detection and recovery. The proposed protocol achieves fast failure detection. However, failure recovery requires a fast detour path setup and relies on a good QNL selection criteria. Each router in the path setup protocol has no knowledge of the QNL in the selected next router. If the QNL is empty, a

Negative Message may be returned and cause a rewinding of the search. This kind of path setup rewinding should be limited by use of good selection criteria. One way to decrease the possibility of path setup rewinding is to increase the information available to each router, for example, the past history of previous path setup or QNL of the neighbors. To make this information available to each router may slow down the path setup in another respect. Therefore, future work on this protocol should include finding good QNL selection criteria.

Multiple failure is another issue not addressed in the protocol. There may be multiple routers detecting different failures more or less simultaneously. It is not a good idea to have multiple path repair processes running at the same time. Simultaneous repairs may result in redundant work and even incorrect path repair due to interference. This is also an open area to be addressed.

CHAPTER 4
KEY ASSIGNMENT FOR ACCESS CONTROL POLICY EXCEPTIONS

In the previous two chapters, two static and one dynamic interdomain access control protocols are discussed. All of them intend to build a secure communication path, though their approaches are different. There is a preliminary requirement for building a secure path, i.e., safe session key distribution. In Chapter 2, we assume that a symmetric key encryption algorithm is used for this purpose. Therefore, every policy gateway should share a secret key with each host within its domain and every two adjacent (neighbor) policy gateways also should share a secret key. For policy gateways in a big domain or in a topological critical location, the burden of managing hundreds (even thousands) of keys may affect their performance.

In this chapter, a cryptographic key assignment scheme is proposed that can be used to assign keys to internetwork domains for safe session key distribution. Instead of remembering hundreds of keys, each domain gateway only needs to remember two keys. We concentrate the discussion on the key assignment scheme itself in this chapter. How to apply the scheme to interdomain access control will be addressed in Chapter 5.

4.1 Background

If system users are divided into distinct groups or classes, $C = \{C_1, C_2, \ldots, C_n\}$, each class should have an encryption key to protect data owned by the users in this class. Each class in the system would have a different clearance and, consequently, a different accessible set. In such a system, the accessible set of a class C_i is the set of classes whose encrypted data can be decrypted by C_i. An access control policy of a system defines the accessible sets of all classes. The objective of this chapter is to design a key assignment scheme so that the access control policy can be enforced.

The most straightforward implementation of this problem requires classes to memorize all encryption keys of their accessible sets. This approach scales poorly. Akl and Taylor first proposed an elegant key assignment scheme [1] to overcome this dilemma, but their scheme only enforces policies in a hierarchical structure. An access control policy expressed in a hierarchy defines the set C as a partially ordered set (poset) on an accessible relation. We will describe the properties of a poset in Section 4.2 and their key assignment scheme in Section 4.5.1. There are many papers [15, 22, 9, 8, 23, 14, 11, 33, 27, 31, 30, 28] that address key assignment for access control in a hierarchy (poset) structure. However, in practice, some systems may need more complex access control policies that can not be expressed in a hierarchy. These complex access control policies may violate the antisymmetric and transitive properties of posets.

For example, suppose a system contains raw data owned by a class C_3 that should remain secret from ordinary users in a class C_1. A special class C_2 exists of users who can preprocess the raw data and translate the data into another form before presenting them to class C_1 users. Namely, C_1 can access C_2 and C_2 can access C_3, but C_1 cannot access C_3. This transitive exception policy is very common and important for real systems.

Moreover, the antisymmetric property of posets forces all classes in an accessible cycle to be equivalent, placing too many restrictions on a system's group/class partition. In Section 4.3, a distributed static database example is provided to show the importance of transitive and antisymmetric exceptions in real practice. Therefore, a policy model based on a hierarchy structure with exceptions, and a new key assignment scheme are proposed. These will enforce access control policies in which antisymmetric and transitive exceptions are included, in addition to the policies with partial ordered set (poset) properties. Compared to schemes for hierarchies, the cost to achieve our more powerful scheme for hierarchies with exceptions is one more key for each user class to memorize or one more step to access its own data.

This chapter is organized as follows: Section 4.2 presents the access control policies that can be handled in the hierarchy structure and in the hierarchy structure with exceptions. Section 4.3 provides an interesting distributed static database example whose access control policy has antisymmetric and transitive exceptions. Section 4.4 discusses the related works with respect to several criteria. All of the existing schemes assume the hierarchy structure. Section 4.5 gives a brief description of Akl and Taylor's scheme. Section 4.6 describes the new cryptographic key assignment in a hierarchical structure with exceptions. Finally, Section 4.7 summarizes this chapter.

4.2 Access Control Policies

An access control policy for a system specifies rules that regulate the information flow between different user classes in the system. In this section, we show that access control policies in the hierarchical structure with exceptions are a superset of those in the hierarchy. To describe the access control policies in both structures, first some definitions are necessary.

Definition 4.2.1 *An n-system is a system with n distinct user classes* $C = \{C_1, C_2, \ldots, C_n\}$.

Definition 4.2.2 *Three properties must hold for an n-system to be a poset on a relation R:*

1. *Reflexive property:* C_iRC_i, *where* $C_i \in C$.
2. *Antisymmetric property:* C_iRC_j *and* $C_jRC_i \Rightarrow C_i = C_j$, *where* C_i *and* $C_j \in C$.
3. *Transitive property:* C_iRC_j *and* $C_jRC_k \Rightarrow C_iRC_k$, *where* C_i, C_j *and* $C_k \in C$.

Definition 4.2.3 *An accessible set of a class C_i in an n-system is the set $AS_i = \{C_j \in C : C_i \text{ can access } C_j\}$.*

Definition 4.2.4 *A dominating set of a class C_i in an n-system is the set $DS_i = \{C_j \in C : C_j \text{ can access } C_i\} = \{C_j \in C : C_i \in AS_j\}$.*

Definition 4.2.5 *An access control policy of an n-system is the collection of all accessible sets $\{AS_1, AS_2, \ldots, AS_n\}$.*

A hierarchical structure for an n-system preserves the three properties of posets on the accessible relation. Thus, any access control policy $\{AS_1, AS_2, \ldots, AS_n\}$ in a hierarchy has following properties: For $i, j, k = 1, 2, \ldots, n$

1. $C_i \in AS_i$ (reflexive property)
2. If $C_j \in AS_i$ and $C_i \in AS_j \Rightarrow C_i = C_j$ (antisymmetric property)
3. If $C_j \in AS_i$ and $C_k \in AS_j \Rightarrow C_k \in AS_i$ (transitive property)

On the other hand, any access control policy in the hierarchy structure with exceptions for an n-system only needs to satisfy two properties:

1. $C_i \in AS_i$ (reflexive property)
2. If $AS_i = AS_j$ and $DS_i = DS_j \Rightarrow C_i = C_j$. (equivalence property)

The combination of reflexive and antisymmetric properties in the hierarchy structure also implies the equivalence property.

Proposition 4.2.1 *If an access control policy has the reflexive and antisymmetric properties, then it also has the equivalence property.*

Proof:

Assume that an access control policy P does not have the equivalence property. That is,

$\exists i, j$ such that $AS_i = AS_j$ and $DS_i = DS_j$, but $C_i \neq C_j$.

By the assumed reflexive property, $C_i \in AS_i$ and $C_j \in AS_j$, so we have $C_i \in AS_j$ and $C_j \in AS_i$, but $C_i \neq C_j$. Thus, P does not have the antisymmetric property.

Any hierarchical access control policy satisfies reflexive, antisymmetric, and transitive properties. By Proposition 4.2.1, this access control policy also satisfies reflexive and equivalence properties and must belong to the hierarchy structure with exceptions. Therefore, the set of hierarchical access control policies with exceptions is a superset of the set of hierarchical access control policies.

Both the antisymmetric and equivalence properties are rules to decide the equivalence of two classes. In terms of the conditions for equivalence testing, the equivalence property is more relaxed than the antisymmetric property. This relaxation is because the absence of transitivity in the hierarchy structure with exceptions makes possible different accessible and dominating sets for two classes, though they can access each other. The relaxation gives a system the opportunity to have a more flexible class/group partition, i.e., two classes that can access each other no longer need to be equivalent. This antisymmetric exception is extremely useful for a system handling indirect remote accesses in a distributed environment.

In order to get a clearer view of these two policy structures, a directed graph (DG) representation of access control policies in both structures is discussed below. A node in a DG represents a class in a system. Two kinds of edges, positive and negative edges, are used in the DG. The positive edges indicate that the source nodes are allowed to access the destination nodes, but the negative edges indicate that the accesses are prohibited. If there is a conflict between these two kinds of edges, negative edges take precedence. The other difference between them is that positive edges have transitive property but negative edges do not. That is, a node A can access another node B if there exists a positive path (sequence of positive edges) from A to B. Whereas the negative path (sequence of negative edges) from A to B does not mean A cannot access B if the path length is greater than one. Basically there are two conditions to decide the accessibility for node A to node B.

Condition 1: If there exists a positive path (a sequence of positive edges) from A to B.

Condition 2: If there exists a negative edge from A to B.

If Condition 1 is true, but not Condition 2, then A can access B. On the other hand, if Condition 1 is false or Condition 2 is true, then A can not access B. Having this kind of DG in mind, it is possible to discuss the policy structures. For the hierarchy structure, the DGs must obey the three properties of posets.

1. Reflexive Property: Each node should have a positive edge to itself. It means each class can access its own data.
2. Transitive Property: A node can access another node if there exists a positive path to that node.
3. Antisymmetric Property: Two nodes are equivalent if both of them have a positive path to the other. In general, a positive path cycle in a DG means all nodes in this cycle can access each other and should be the same class. Thus, for an n-system, the n node DG must be acyclic.

Obviously, the negative edges do not exist in hierarchies, since their purpose is to kill the transitive property selectively. They are used only in the hierarchy structure with exceptions. For the hierarchy structure with exceptions, the DGs have two properties.

1. Reflexive Property: Each node should have a positive edge to itself.
2. Equivalence Property: Two nodes are equivalent if both of them have a positive path to the other and the same immediate predecessors and successors for the negative edges.

Two access control policy examples that can be handled in the hierarchy structure with exceptions, but not in the hierarchy structure are given in Figure 4.1 and Figure 4.2. For clarity, self-pointing edges for the reflexive property in the DG policy representation are omitted for rest of the chapter.

class	accessible set	dominating set
C_1	C_1, C_2	C_1
C_2	C_2, C_3	C_1, C_2
C_3	C_3	C_2, C_3

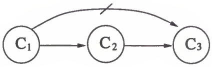


Figure 4.1. A policy in a 3-system that only violates the transitive property

class	accessible set	dominating set
C_1	C_1, C_2, C_3	C_1, C_3
C_2	C_2, C_3	C_1, C_2
C_3	C_1, C_3	C_1, C_2, C_3

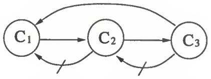


Figure 4.2. A policy in a 3-system that violates the antisymmetric and transitive properties

In Figure 4.2, there is a positive cycle $C_1 \to C_2 \to C_3 \to C_1$. These three classes are not equivalent in the hierarchy structure with exceptions because $AS_i \neq AS_j$ and $DS_i \neq DS_j$ for $i \neq j$ and $1 \leq i, j \leq 3$. Two transitive exceptions also exist in this example. One is $C_2 \to C_3$ and $C_3 \to C_1$, but $C_2 \not\to C_1$. The other is $C_3 \to C_1$ and $C_1 \to C_2$, but $C_3 \not\to C_2$. These two transitive exceptions allow different accessible sets and dominating sets for these three classes. Therefore, an access control policy with the antisymmetric exceptions should also have transitive exceptions.

An access control policy $\{AS_1, AS_2, \ldots, AS_n\}$ of an n-system can be represented as an $n \times n$ matrix A, where

$$A_{ij} = \begin{cases} 1 & \text{if } C_j \in AS_i \\ 0 & \text{otherwise} \end{cases}$$

For example, the access control policy in Figure 4.2 can be represented as Table 4.1. This

Table 4.1. Matrix A – a policy in a 3-system

	C_1	C_2	C_3
C_1	1	1	1
C_2	0	1	1
C_3	1	0	1

matrix A is different from the access control matrix (ACM) of HRU model [10]. The ACM is a matrix correlating the subject, object and the authorizations owned by each subject on each object. The rows of ACM correspond to subjects and the columns to the objects. Each element ACM$[i, j]$ contains the access modes for which the subject i is authorized on object j. However, the matrix used here merges subjects and objects with the same security level into a class. Both the rows and columns of the matrix A correspond to the classes. The element A_{ij} of matrix A denotes the access authorization between classes C_i and C_j. $A_{ij} = 1$ indicates that the class C_i has the privilege to access the class C_j; $A_{ij} = 0$ otherwise.

4.3 Policy Exception Example: A Distributed Static Database System

Consider a static database that contains some tables with confidential records. Users can only make a request to a specific query processor to retrieve information in these tables. Table 4.2 EMPLOYEE gives an example of confidential records in a static database system. A user in this system can not directly access the confidential records. Instead he or she can make requests such as "How many employees are ranked 1 ?" or "How many employees have salaries of $60,000 or higher ?" to the query processor. The query processor goes over

Table 4.2. Confidential records for EMPLOYEE

Name	Rank	Sex	Age	Salary
Joe	1	M	50	60,000
Sam	2	M	45	55,000
Mary	1	F	46	62,000
John	3	M	36	50,000

the EMPLOYEE table and answers the user: "There are two employees are ranked 1" and "There are two employees with salaries 60,000 or higher".

In this context, there are three classes: C_1 for users, C_2 for the query processor, and C_3 for the EMPLOYEE table. The EMPLOYEE table is encrypted to prevent direct access from C_1. C_1 makes a request to C_2, and C_2 derives the encryption key of C_3 to decrypt the EMPLOYEE table. C_2 sends the answer encrypted by its own encryption key to C_1. Then C_1 decrypts it and gets the answer. A transitive exception exists among these three classes. That is, C_1 can access C_2, C_2 can access C_3, but C_1 cannot access C_3. Encryption of the answer is necessary whenever there are several classes of users and each user class only can retrieve the static information over some confidential tables through a different query processor. For simplicity, we assume there is only one user class and one query processor.

Next, the example is extended to a distributed environment to show that the anti-symmetric exception is also important. Suppose a company has distributed sites connected by a network. Each site has its own confidential tables and three classes as above. A user who wants remote access to the static information from another site is not allowed to make requests directly to the remote query processor. The request must go to the local query processor first, which decides whether to forward the request to the remote query processor. If the request is forwarded, the remote query processor prepares and encrypts the answer and sends it back to the local query processor. The local query processor first decrypts the answer, then re-encrypts the answer so that the user can decrypt it. The two-site example is depicted in Figure 4.3 (Eight negative edges representing the transitive exceptions are omitted here for clarity). The corresponding access control policy for this example is shown in Table 4.3.

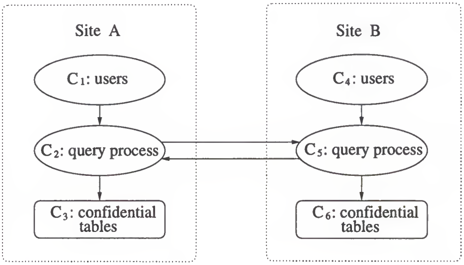


Figure 4.3. A two-site distributed static database system

Table 4.3. Matrix A – the access control policy for a two-site distributed static database system

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	1	0	0	0	0
C_2	0	1	1	0	1	0
C_3	0	0	1	0	0	0
C_4	0	0	0	1	1	0
C_5	0	1	0	0	1	1
C_6	0	0	0	0	0	1

In this example, both query processors in sites A and B can access each other, but they are not equivalent. There are eight transitive exceptions and one antisymmetric exception in this example. We will show that this example can be handled by our key assignment scheme in Section 4.6, but not by other schemes in the literature.

4.4 Related Work

The cryptographic key assignment problem for a system to enforce the access control policy was first investigated by Akl and Taylor [1]. They proposed an elegant solution to enforce access policies modeled as a partial order set. Their scheme is simple with respect to the key generation and derivation procedures, but suffers high memory overhead, weak appendability and restricted (hierarchy-based) set of policies. Appendability weakness means that the keys for existing classes need to be recomputed when a new security class is added. For a decade, research on this key assignment problem concentrated on finding solutions to reduce memory overhead and increase appendability. As the computer technology is evolving toward distributed computing, the hierarchy policy structure becomes too restricted to model the needs for distributed systems. This chapter intends to develop a key assignment scheme that can enforce a richer set of policies for distributed systems.

Mackinnon et al. [15] presented an improved scheme using canonical assignment to reduce memory overhead. Harn and Lin [9] proposed another scheme in which the key assignment is handled by a bottom-up approach, while [1] and [15] are top-down approaches. Appendability weakness is addressed by Sandhu [22], Harn, Chien and Kiesler [8], Liaw [14], Hwang et al. [11], Wu et al. [33], Tsai et al. [27]. This appendability problem is difficult to solve and no existing scheme has achieved complete appendability. Some solutions to reduce the complexity are to limit the access control policy to a tree structure, which is a special case of the hierarchy structure. Others reduce the recomputation to a smaller set of existing keys with different degrees of compensation. In [22], an ID-based scheme is presented by iteratively applying one-way functions to solve the problem only for tree like structures. Harn, Chien, and Kiesler [8] presented a scheme that divides a system into several subgroups to reduce the affected keys while a new class is added. Each subgroup has a root class responsible for replying to a request from higher subgroup. Liaw's [14] scheme, which is based on the RSA [20] public key system increases appendability, but it still suffers from key recomputation overhead when the added class has a high security level. Hwang et al [11] presented a scheme in which the addition/deletion of a class needs only to modify the keys of immediate higher classes. In their scheme, a class with r immediate lower classes must assign a unique integer from 1 to r to each immediate lower classes. In order to derive the immediate lower classes' keys, each class needs to memorize the integer identifiers of them. The scheme of Wu et al [33] is based on the Chinese remainder theorem and it achieves the appendability without changing other existing keys, but a set of public parameters needs to be modified. Tsai et al [27] proposed a scheme based on Rabin's public key system and the concept of the Chinese Remainder theorem and Newton's interpolating method. Their scheme still needs to update the secret interpolating polynomials of those security classes that have access privileges on a new class, though no existing keys need to be modified.

The other criterion to evaluate a scheme is the efficiency of key derivation. Some schemes [1], [15], [9], [14], [33], [27] derive a subordinate key directly, but others [22], [8], [11] need iterative derivation.

Although the new scheme proposed in this chapter has the same advantages and disadvantages as in [1], a new and important capability for our scheme that broadens access control from the hierarchical structure to the hierarchy structure with exceptions is never addressed before.

4.5 Akl and Taylor's Key Assignment Scheme

4.5.1 Key Assignment and Derivation

In Akl and Taylor's key assignment scheme, there is a central authority C_0. In an n-system, C_0 must generate a set of encryption keys $K_1, K_2, \ldots, K_n$ and send K_i to each class C_i securely. Each class C_i can use its own key K_i and some public information T_i and T_j to derive the class C_j's key K_j if C_i is allowed to access C_j. C_0 proceeds the following steps.

1. Generate a secret number K_0.

2. Compute the product M of two large prime numbers and make it public.

For $i, j = 1, 2, \ldots, n$

3. Assign a distinct prime number P_i for each class C_i.

4. Compute public information $T_i = \prod_{A_{ij}=0} P_j$, where matrix A is defined in Section 4.2.

5. Assign each $K_i = K_0^{T_i} \bmod M$.

The class C_i uses the key derivation function,

$$f(K_i, T_i, T_j) = K_i^{T_j/T_i} \bmod M,$$

to derive C_j's key K_j, where

$$K_j = K_0^{T_j} = (K_0^{T_i})^{T_j/T_i} = K_i^{T_j/T_i} \bmod M.$$

The derivation function $f(K_i, T_i, T_j)$ is feasible to compute if and only if T_j/T_i is an integer (see proof in next section). This statement relies on the belief [20] that computing r-th roots (mod M) for integral $r > 1$ is as difficult as factoring M. Their scheme assigns all public information T's in such a way that T_j/T_i is an integer if and only if the policy allows C_i to access C_j.

For example, the policy of a 5-system in Table 4.4 (the corresponding DG policy representation in Figure 4.4) can be enforced by assigning keys:

$$K_1 = K_0 \bmod M$$
$$K_2 = K_0^2 \bmod M$$
$$K_3 = K_0^{2\cdot 3\cdot 7} \bmod M$$
$$K_4 = K_0^{2\cdot 3\cdot 5} \bmod M$$
$$K_5 = K_0^{2\cdot 3\cdot 5\cdot 7} \bmod M.$$

Note that there are no transitive or antisymmetric exceptions in this example. Their scheme can only enforce policies in the hierarchy.

Table 4.4. Akl and Taylor's scheme for a policy in a 5-system

	C_1	C_2	C_3	C_4	C_5	P_i	T_i
C_1	1	1	1	1	1	2	1
C_2	0	1	1	1	1	3	2
C_3	0	0	1	0	1	5	$2\cdot 3\cdot 7$
C_4	0	0	0	1	1	7	$2\cdot 3\cdot 5$
C_5	0	0	0	0	1	11	$2\cdot 3\cdot 5\cdot 7$

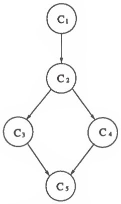


Figure 4.4. DG representation of a policy in a 5-system

4.5.2 Correctness of the Akl and Taylor's Scheme

A correct key assignment scheme should prohibit all illegal key derivations. There are two kinds of illegal key derivations in the hierarchy structure:

1. A class C_i derives a class C_j's key, but the policy forbids C_i from accessing C_j.
2. A set of classes $H \subset C$ collusively derive a class C_j's key, but the policy does not allow any class in H to access C_j.

In Akl and Taylor's paper, they give a theorem (below) and several propositions (4.5.1, 4.5.4, 4.5.5, and 4.5.6) to show the correctness of their scheme. We add two propositions (4.5.2 and 4.5.3) here that make the proof of correctness more complete. (The notation $T_i \mid T_j$ below means that T_i is a factor of T_j and $T_i \nmid T_j$ otherwise).

Theorem [1] *Let t and $t_1, \ldots, t_n$ be given integers and suppose there is a feasibly computable function G for which*

$$K^t = G(K^{t_1}, K^{t_2}, \ldots, K^{t_n})(\text{mod} M)$$

for every K in Z_M^X, the group of units mod M. Let

$$d = gcd\{t_i\},\ e = gcd\{t, d\},\ and\ r = d/e.$$

Then we can feasibly compute r-th roots in Z_M^X.

Proof:

Taking any H in Z_M^X, we will compute $H^{1/r}(\text{mod} M)$. Let $K = H^{1/d}$ (we cannot necessary compute K) and $r_i = t_i/d$. Choose a and b so that $e = at + bd$. Then

$$\begin{aligned} H^{1/r} &= H^{e/d} = K^e = K^{bd}K^{at} = H^bG(K^{t_1}, \ldots, K^{t_n})^a \\ &= H^bG(K^{dr_1}, \ldots, K^{dr_n})^a \\ &= H^bG(H^{r_1}, \ldots, H^{r_n})^a(\text{mod} M), \end{aligned}$$

and this can feasibly be computed.

The argument relies on the current belief [20, pp. 126] that computing r-th roots (mod M) for integral $r > 1$ is as difficult as factoring M. The case $r = 2$ is proved in [17]. The method therein generalizes to the case $r \mid \Phi(M)$, where $\Phi(M)$ is the Euler totient function. In Akl and Taylor's paper, Proposition 4.5.1 only has the "only if" part. Because the value T_j/T_i can be very large, the function $f = K_i^{T_j/T_i} \bmod M$ is not feasible to compute. In order to make the "if" part of this proposition also true, we assume that there exists an upper bound U for the value T_j/T_i so that f is feasible to compute.

Proposition 4.5.1 *The key derivation function $f(K_i, T_i, T_j)$ is feasible to compute iff $T_i \mid T_j$.*

Proof:

If $f(K_i, T_i, T_j)$ is feasible to compute $\Rightarrow K_j = ((K_0)^{T_j})(\text{mod} M)$ is feasible to compute from K_i for every K_0.

By the Theorem above, $r = 1$ and hence $T_i \mid T_j$. Otherwise we could compute nontrivial roots in Z_M^X.

Proposition 4.5.2 *K_j can be derived from K_i iff $f(K_i, T_i, T_j) = (K_i)^{T_j/T_i} = K_j$ is feasible to compute.*

Proof:

If $f(K_i, T_i, T_j)$ is feasible to compute, then it is obvious that K_j can be derived from K_i.

Conversely, if $f(K_i, T_i, T_j) = (K_i)^{T_j/T_i} = K_j$ is infeasible to compute $\Rightarrow T_j/T_i$ is not an integer. (by Proposition 4.5.1)

Suppose that K_j can be derived from K_i by using other public information X and Y so that $f(K_i, X, Y) = K_j$ is feasible to compute $\Rightarrow Y/X$ is an integer. (by Proposition 4.5.1)

Since $f(K_i, X, Y) = ((K_0)^{T_i})^{Y/X} = K_j \Rightarrow T_i \cdot Y/X = T_j \Rightarrow Y/X = T_j/T_i$ - contradiction.

Thus K_j cannot be derived from K_i.

Proposition 4.5.3 *K_j can be derived from K_i iff $T_i \mid T_j$*

Proof:

By Propositions 4.5.1 and 4.5.2, the result follows.

Proposition 4.5.4 *Under the assignment scheme in Section 4.5.1, $T_i \mid T_j$ iff $A_{ij} = 1$.*

Proof:

If $A_{ij} = 1 \Rightarrow C_j \in AS_i$. For any $C_k \in AS_j$, we have $C_k \in AS_i$ (transitive property) $\Rightarrow AS_j \subseteq AS_i$.

Since $T_i = \prod_{A_{ik}=0} P_k = \prod_{C_k \notin AS_i} P_k$, $T_j = \prod_{A_{jk}=0} P_k = \prod_{C_k \notin AS_j} P_k$, and $AS_j \subseteq AS_i \Rightarrow T_i \mid T_j$.

Conversely if $A_{ij} = 0 \Rightarrow P_j \mid T_i$. Since $A_{jj} = 1$ (reflexive property) $\Rightarrow P_j \not| T_j \Rightarrow T_i \not| T_j$.

Proposition 4.5.5 *Under the assignment scheme in Section 4.5.1, a key K_j can be feasibly computed from a set of keys $\{K_i : C_i \in H\}$ iff $\gcd(T_i : C_i \in H) \mid T_j$, where $H \subset C$.*

Proof:

If $g = \gcd(T_i : C_i \in H)$, then we can choose integers a_i such that $g = \sum_H a_i T_i$. If $T_j = gm$ for some integer m, then $K_j = (K_0)^{T_j} = (K_0)^{gm} = \prod_H (K_0)^{T_i a_i m} = \prod_H (K_i)^{a_i m}$ and K_j can be computed from K_i's.

Conversely, if there exists a feasibly computable function for obtaining $K_j = (K_0)^{T_j} (\mathrm{mod}\ M)$ from the K_i's for every K_0. By the Theorem above, $r = 1$ and $\gcd(T_i : C_i \in H) \mid T_j$.

Proposition 4.5.6 *Under the assignment scheme in Section 4.5.1, a set of classes $H \subset C$ can not collusively derive a class C_j,'s key if $A_{ij} = 0$ for all $T_i \in H$.*

Proof:

$P_j \mid T_i$ whenever $A_{ij} = 0$, therefore, $P_j \mid \gcd(T_i : A_{ij} = 0)$.

But $A_{jj} = 1$ (reflexive property) $\Rightarrow (P_j \not| T_j) \Rightarrow \gcd(T_i : A_{ij} = 0) \not| T_j$.

By Proposition 4.5.5, the result follows.

From Propositions 4.5.3 and 4.5.4, K_j can be derived from K_i if and only if the policy allows C_i to access C_j. Thus, the first illegal key derivation is prohibited. From Propositions 4.5.5 and 4.5.6, the second illegal key derivation is also prohibited.

4.6 The New Key Assignment Scheme

The new key assignment scheme basically follows the same approach as Akl and Taylor's scheme. One of the differences is that there are two keys (K^e and K^d) assigned to each class: K^e for data encryption and K^d for key derivation. If a class C_i would like to access data of a class C_j, C_i uses its own derivation key K_i^d and some public information to derive C_j's encryption key K_j^e. The purpose of assigning two keys for each class is to enforce transitive exception policies. For example, a transitive exception policy for a 3-system $\{C_1, C_2, C_3\}$ defines that C_1 can access C_2 and C_2 can access C_3, but C_1 cannot access C_3 as in Figure 4.5. If only one key is assigned to each class, there is no way to enforce the transitive exception because C_1 can always derive C_3's key by deriving C_2's key first. With two keys for each class in our assignment scheme, this problem can be solved. The idea of enforcing transitive exceptions in the assignment scheme is shown in Figure 4.5 (here, an arrow in the key derivation rule means the source key can derive the destination key). Furthermore, two keys for each class also make it possible to enforce antisymmetric exceptions. Therefore, two classes can access each other's information without being equivalent. Figure 4.6 shows the key derivation rule for enforcing antisymmetric exceptions. The key assignment scheme proposed in this section assigns keys in a way that follows the key derivation rules so that the transitive and antisymmetric exceptions can be enforced.

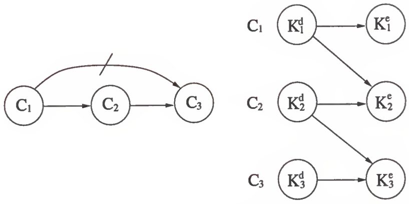


Figure 4.5. A 3-system with a transitive exception and the corresponding key derivation rule

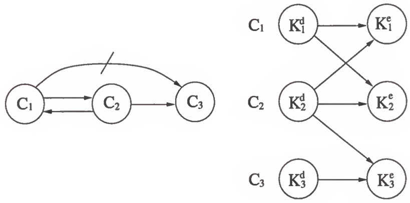


Figure 4.6. A 3-system with an antisymmetric exception and the corresponding key derivation rule

4.6.1 Key Assignment and Derivation

The access control policy in a matrix A, described in Section 4.2, only indicates which class can access another. Transitive exception information is embedded in A. Therefore, an algorithm is necessary to translate the matrix A into another $n \times n$ matrix UM that indicates explicitly the transitive exception information, where

$$UM_{ij} = \begin{cases} 1 & \text{if } A_{ij} = 1 \\ -1 & \text{if } \exists\, k_1, \exists\, k_2, \ldots, \exists\, k_n, \\ & \text{such that } 1 \le k_i \le n, \\ & \text{for all } i,\ 1 \le m \le n-2, \\ & A_{ik_1} = A_{k_1k_2} = \ldots = A_{k_mj} = 1 \\ & \text{and } A_{ij} = 0.\text{(transitive exception)} \\ 0 & \text{otherwise} \end{cases}$$

The transformation algorithm computes the transitive closure $A^\star$ of A and compares $A^\star$ and A to find out the transitive exceptions so that the matrix UM can be constructed.

Algorithm 6.1 *Transformation from A to UM using* $A^\star$ = *transitive closure* (A)
$UM = A - (A^\star - A) = 2A - A^\star$.

We demonstrate the transformation of algorithm 6.1 with the distributed static database example in Table 4.3. Table 4.5 is the resulting matrix UM. With the transformed matrix

Table 4.5. Matrix UM – a policy in a 6-system with explicit transitive exception information

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	1	-1	0	-1	-1
C_2	0	1	1	0	1	-1
C_3	0	0	1	0	0	0
C_4	0	-1	-1	1	1	-1
C_5	0	1	-1	0	1	1
C_6	0	0	0	0	0	1

UM, the central authority C_0 proceeds as followings.

1. Generate a secret number K_0.

2. Compute the product M of two large prime numbers and make it public.

For $i, j, k = 1, 2, \ldots, n$

3. Assign a distinct prime number P_i to each row UM_i of UM and $P_{i'}$ to each row UM_i of UM where

$$P_{i'} = \begin{cases} \text{another distinct prime number} & \text{if } \exists\, j \text{ such that } UM_{ij} = -1 \\ 1 & \text{otherwise} \end{cases}$$

4. Compute the public information

$$T_i^d = (\prod_{UM_{ij}=0} P_j) \cdot P_{i'}$$

$$T_i^e = (\prod_{UM_{ij}=0} P_j) \cdot (\prod_{UM_{ki}=1} P_{k'})$$

5. Assign keys

$$K_i^d = (K_0)^{T_i^d} \bmod M$$

$$K_i^e = (K_0)^{T_i^e} \bmod M$$

to each class C_i and send them securely.

The class C_i can use the key derivation function

$$f(K_i^d, T_i^d, T_j^e) = (K_i^d)^{T_j^e/T_i^d} \bmod M$$

to derive the encryption key K_j^e of C_j, where

$$\begin{aligned} K_j^e &= (K_0)^{T_j^e} \bmod M \\ &= ((K_0)^{T_i^d})^{T_j^e/T_i^d} \bmod M \\ &= (K_i^d)^{T_j^e/T_i^d} \bmod M \end{aligned}$$

The way to assign public information T's in step 2 ensures that T_j^e/T_i^d is an integer if and only if the policy allows C_i to access C_j. The key derivation function $f(K_i^d, T_i^d, T_j^e)$ is feasible to compute if and only if T_j^e/T_i^d is an integer (Proposition 4.5.1). Therefore, C_i can access C_j's data by deriving C_j's encryption key using the key derivation function $f(K_i^d, T_i^d, T_j^e)$.

For the access control policy in Table 4.5, Figure 4.7 shows the corresponding key derivation rule and Table 4.6 illustrates a possible assignment of the prime numbers and public information. The key assignments using the new scheme are:

$K_1^d = K_0^{7\cdot17} \bmod M$ $\quad$ $K_1^e = K_0^{7\cdot17} \bmod M$

$K_2^d = K_0^{2\cdot7\cdot19} \bmod M$ $\quad$ $K_2^e = K_0^{2\cdot7\cdot17\cdot19\cdot29} \bmod M$

$K_3^d = K_0^{2\cdot3\cdot7\cdot11\cdot13} \bmod M$ $\quad$ $K_3^e = K_0^{2\cdot3\cdot7\cdot11\cdot13\cdot19} \bmod M$

$K_4^d = K_0^{2\cdot23} \bmod M$ $\quad$ $K_4^e = K_0^{2\cdot23} \bmod M$

$K_5^d = K_0^{2\cdot7\cdot29} \bmod M$ $\quad$ $K_5^e = K_0^{2\cdot7\cdot19\cdot23\cdot29} \bmod M$

$K_6^d = K_0^{2\cdot3\cdot5\cdot7\cdot11} \bmod M$ $\quad$ $K_6^e = K_0^{2\cdot3\cdot5\cdot7\cdot11\cdot29} \bmod M$

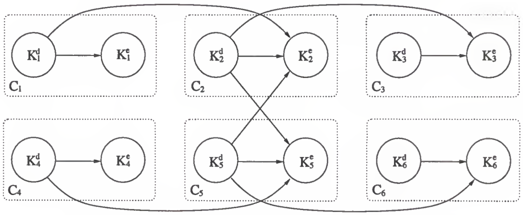


Figure 4.7. The key derivation rule for the policy in Table 4.5

4.6.2 Correctness of the Proposed Scheme

There are three possible illegal key derivations in the hierarchy structure with exceptions:

1. A class C_a derives class C_b's keys, but the policy does not allow C_a to access C_b.

Table 4.6. Prime numbers and public information for the policy in Table 4.5

	P_i	P_i'	T_i^d	T_i^e
C_1	2	17	7·17	7·17
C_2	3	19	2·7·19	2·7·17·19·29
C_3	5	1	2·3·7·11·13	2·3·7·11·13·19
C_4	7	23	2·23	2·23
C_5	11	29	2·7·29	2·7·19·23·29
C_6	13	1	2·3·5·7·11	2·3·5·7·11·29

2. A class C_a derives class C_c's keys by iteratively deriving another classes C_{b_1}'s, C_{b_2}'s, $\ldots, C_{b_m}$'s keys, where the policy allows C_a to access C_{b_1}, C_{b_1} to access $C_{b_2}, \ldots, C_{b_m}$ to access C_c, but not C_a to access C_c.
3. A set of classes $H \subset C$ collusively derive class C_a's keys, but the policy does not allow any class in H to access C_a.

We use indices $a, b, c \in \{1, 2, \ldots, n\}$ along with $i, j, k \in \{1, 2, \ldots, n\}$ (used in assignment definitions in Section 4.6.1) to avoid confusion. The proposed key assignment scheme is correct and the above three illegal key derivations are infeasible. Lemmas 4.6.6, 4.6.7 and 4.6.8 (below) prove the first, second, and third illegal key derivations respectively are infeasible.

First, we give a proposition showing the key derivation relationship between keys.

Proposition 4.6.1 *Under the key assignment scheme in Section 4.6.1, if a key K_a^x can derive a key K_b^x (for $x = d$ or e), then (i) any key can be derived from K_b^x, it also can be derived from K_a^x; (ii) any key can derive K_a^x, it also can derive K_b^x.*

Proof:

(i) Let a set of keys $A = \{K_c^x : K_c^x$ can be derived from $K_a^x\}$ and a set of keys $B = \{K_c^x :\ K_c^x$

can be derived from $K_b^x\}$.

By Proposition 4.5.3, we can rewrite $A = \{K_c^x : T_a^x \mid T_c^x\}$ and $B = \{K_c^x : T_b^x \mid T_c^x\}$.

Since K_a^x can derive $K_b^x \Rightarrow T_a^x \mid T_b^x$ (by Proposition 4.5.3) $\Rightarrow A \supseteq B$.

(ii) Let a set of keys $A = \{K_c^x : K_c^x$ can derive $K_a^x\}$ and a set of keys $B = \{K_c^x :\ K_c^x$ can derive $K_b^x\}$.

By Proposition 4.5.3, we can rewrite $A = \{K_c^x : T_c^x \mid T_a^x\}$ and $B = \{K_c^x : T_c^x \mid T_b^x\}$.

Since K_a^x can derive $K_b^x \Rightarrow T_a^x \mid T_b^x$ (by Proposition 4.5.3) $\Rightarrow A \subseteq B$.

In the key assignment scheme in Section 4.6.1, each class can use its own derivation key to derive its own encryption key. Because of this property, each class only needs to memorize a derivation key, but one more key derivation would be necessary whenever it wants to access its own data. The Proposition 4.6.2 shows this property.

Proposition 4.6.2 *Under the key assignment scheme in Section 4.6.1, K_a^d can always derive K_a^e.*

Proof:

By Proposition 4.5.3, we only need to show $T_a^d \mid T_a^e$ is always true.

$$T_a^e / T_a^d = \frac{(\prod_{UM_{aj}=0} P_j)\cdot(\prod_{UM_{ka}=1} P_{k'})}{(\prod_{UM_{aj}=0} P_j)\cdot P_{a'}}$$

$$= \frac{(\prod_{UM_{ka}=1} P_{k'})}{P_{a'}}$$

$$= \prod_{UM_{ka}=1, k\neq a} P_{k'}$$

The last inference step is because $UM_{aa} = 1$ is always true at index $k = a$.

Therefore, T_a^e / T_a^d is always an integer and we have the result.

Proposition 4.6.3 *Under the key assignment scheme in Section 4.6.1, (i) if a key can be* *derived from K_a^e, it also can be derived from K_a^d; (ii)if a key can derive K_a^d, it also can derive K_a^e.*

Proof:

By Propositions 4.6.1 and 4.6.2, the result follows.

Proposition 4.6.4 *If $UM_{ab} = 1$, then $UM_{aj} = 0$ implies $UM_{bj} = 0$.*

Proof:

Case 1: Let $UM_{ab} = 1$, $UM_{aj} = 0$, and $UM_{bj} = 1$.

$UM_{ab} = 1$ and $UM_{bj} = 1 \Rightarrow UM_{aj} = 1$ if $A_{aj} = 1$ (transitivity), or $UM_{aj} = -1$ if $A_{aj} = 0$ (transitive exception).

Thus, $UM_{aj} \neq 0$ - contradiction.

Case 2: Let $UM_{ab} = 1$, $UM_{aj} = 0$, and $UM_{bj} = -1$.

$UM_{bj} = -1 \Rightarrow \exists K_1, K_2, \ldots, K_m$ such that $A_{bk_1} = A_{k_1k_2} = \ldots = A_{k_mj} = 1$ but $A_{bj} = 0$.

$UM_{aj} = 0 \Rightarrow A_{aj} = 0$.

$UM_{ab} = 1 \Rightarrow A_{ab} = 1 \Rightarrow A_{ab} = A_{bk_1} = A_{k_1k_2} = \ldots = A_{k_mj} = 1$ but $A_{aj} = 0$ (transitive exception) $\Rightarrow UM_{aj} = -1$.

Thus, $UM_{aj} \neq 0$ - contradiction.

Since $UM_{xy} \in \{-1, 0, 1\}$ and $UM_{bj} \neq -1$, $UM_{bj} \neq 1$, so $UM_{bj} = 0$

Proposition 4.6.5 *Under the key assignment scheme in Section 4.6.1, $T_a^d \mid T_b^e$ iff $UM_{ab} = 1$.*

Proof:

By Proposition 4.6.4, if $UM_{ab} = 1$,

then ($UM_{aj} = 0 \Rightarrow UM_{bj} = 0$).

Thus, $\frac{\prod_{UM_{bj}=0} P_j}{\prod_{UM_{aj}=0} P_j} = I$, where I is a positive integer.

$$T_b^e / T_a^d = \frac{(\prod_{UM_{bj}=0} P_j)\cdot(\prod_{UM_{kb}=1} P_{k'})}{(\prod_{UM_{aj}=0} P_j)\cdot P_{a'}}$$
$$= I \cdot \frac{\prod_{UM_{kb}=1} P_{k'}}{P_{a'}}$$
$$= I \cdot \prod_{UM_{kb}=1, k\neq a} P_{k'}$$

The last inference step is because $UM_{ab} = 1$ is true at index $k = a$.

Thus, $T_a^d \mid T_b^e$.

Conversely, if $UM_{ab} \neq 1 \Rightarrow$ either (1) $UM_{ab} = 0$ or (2) $UM_{ab} = -1$.

(1) Since $T_a^d = (\prod_{UM_{aj}=0} P_j) \cdot P_{a'}$.

If $UM_{ab} = 0 \Rightarrow P_b \mid T_a^d$

since $UM_{bb} = 1$ and

$T_b^e = (\prod_{UM_{bj}=0} P_j)(\prod_{UM_{kb}=1} P_{k'})$

So $P_b \nmid T_b^e$,

Thus, $T_a^d \nmid T_b^e$.

(2) If $UM_{ab} = -1 \Rightarrow P_{a'} \mid T_a^d$ and $P_{a'} \neq 1$.

But $P_{a'} \nmid T_b^e$,

Thus, $T_a^d \nmid T_b^e$.

Lemma 4.6.6 *If the policy does not allow C_a to access C_b, then it is infeasible for C_a to derive C_b's keys under the key assignment scheme in Section 4.6.1.*

Proof:

The available information for C_a are K_a^d, K_a^e, and all public information T_i^d, and T_i^e for all i.

By Proposition 4.5.2, only four possible inputs (K_a^d, T_a^d, T_b^d), (K_a^d, T_a^d, T_b^e), (K_a^e, T_a^e, T_b^d), and (K_a^e, T_a^e, T_b^e) to function f need to be considered.

By Proposition 4.6.3, we only need to show one input (K_a^d, T_a^d, T_b^e) to f is not feasible to compute.

Since C_a is not allowed to access $C_b \Rightarrow T_a^d \not\mid T_b^e$ (by Proposition 4.6.5) $\Rightarrow f(K_a^d, T_a^d, T_b^e)$ is not feasible to compute. (by Proposition 4.5.1)

If an access control policy of a system contains transitive exceptions, the possibility of the second illegal key derivation becomes a threat. Fortunately, our key assignment scheme is immune from this threat.

Lemma 4.6.7 *If the policy does not allow C_a to access C_c, but C_a can access C_{b_1}, C_{b_1} can access $C_{b_2}, \dots, C_{b_m}$ can access C_c for $1 \leq m \leq n-2$, then it is infeasible for C_a to iteratively derive C_c's keys by deriving C_{b_1}'s, C_{b_2}'s, $\dots, C_{b_m}$'s keys under the key assignment scheme in Section 4.6.1.*

Proof:

Basically we want to show that there does not exist any key derivation chain from K_a^x to K_c^x for $x = d$ or e.

Suppose there exists a key derivation chain $K_a^x \rightarrow K_{b_1}^x \rightarrow K_{b_2}^x \rightarrow \dots \rightarrow K_{b_m}^x \rightarrow K_c^x$.

Since K_a^x can derive $K_{b_1}^x$ and $K_{b_1}^x$ can derive $K_{b_2}^x \Rightarrow K_a^x$ can derive $K_{b_2}^x$ (by Proposition 4.6.1)

By applying Proposition 4.6.1 m times, we have the result that K_a^x can derive K_c^x.

It is a contradiction.

Thus, there does not exist any key derivation chain from K_a^x to K_c^x.

Finally, lemma 4.6.8 shows our key assignment scheme is infeasible for the collusive attack. That is, a set of malicious classes combines their information to derive a key that the policy does not allow them to derive.

Lemma 4.6.8 *If the policy does not allow any class in a set of classes $H \subset C$ to access C_a, then it is infeasible for classes in H to collusively derive C_a's keys under the key assignment* *scheme in Section 4.6.1.*

Proof:

By Proposition 4.6.3, we only need to show a set of derivation keys $\{K_b^d : C_b \in H\}$ can not collusively derive C_a's encryption key K_a^e, where $UM_{ba} \neq 1$.

If $UM_{ba} = 0$, $\exists P_a \mid T_b^d$; or if $UM_{ba} = -1$, $\exists P_{b'} \mid T_b^d$ where $P_{b'} \neq 1$.

Therefore, $P_a \mid \gcd(T_b^d : UM_{ba} = 0)$; or $P_{b'} \mid \gcd(T_b^d : UM_{ba} = -1)$ where $P_{b'} \neq 1$.

But, $P_a \not| T_a^e$ is always true, and $P_{b'} \not| T_a^e \; if \; UM_{ba} = -1$.

Thus, $\gcd(T_b^d : UM_{ba} \neq 1) \not| T_a^e$.

By Proposition 4.5.5, we have the result.

We have proved correctness of the proposed key assignment scheme by showing that the three possible illegal key derivations, listed at the beginning of the section, are not feasible.

4.7 Chapter Summary

This Chapter gives a cryptographic key assignment scheme for controlling access to data in an organization where the access control policy for this organization can be any policy in the hierarchy structure with exceptions. This structure can model not only the access control policies in the hierarchy, but also more complex policies that have antisymmetric or transitive exceptions. Transitive exceptions usually occur in a system that is not hierarchy-based. Both antisymmetric and transitive exceptions are quite common for a distributed system as the example shown in Section 4.3. The cost of this new cryptographic key assignment scheme, compared to the Akl and Taylor's scheme, is one more key for each class to memorize or one more step to access its own data.

There are two keys (derivation and encryption keys) assigned to each class in our cryptographic key assignment scheme. The encryption key for a user class is used to encrypt data to protect against illegal access from other user classes. The derivation key for a user class is used to derive the encryption keys of other user classes in order to decrypt and access their data. The key assignment scheme assures that a class' derivation key cannot derive a class' encryption key in violation of the policy.

CHAPTER 5
INTERDOMAIN SAFE SESSION KEY DISTRIBUTION

In this chapter, we will address how to apply the proposed key assignment scheme to internetwork domain key distribution.

First, we need to know the topology of the current Internet and Section 5.1 gives the brief overview. Second, Section 5.2 discusses how to transfer the Internet topology to a directed graph (DG) defined in Chapter 4 which represents the internetwork domain access control policy. Before performing the transformation, it is necessary to argue the reasonable assumption about access privileges among domains in the current Internet. Section 5.3 demonstrates how to use the assigned keys to distribute the session key. Finally, a summary is given in Section 5.4.

5.1 Current Internet Topology

The current Internet is basically a hierarchical structure with exceptions [6]. It is composed of many different stub domains and transit domains. Stub domains like government, university, and private companies carry only network traffic to and from end systems within their own domain. While transit domains like MAN, LAN, WAN carry network traffic to and from other domains.

The links between WANs form a global transit backbone of the Internet. The links among stubs, LANs and WANs, together with the transit backbone, make the Internet as a connected graph so that any host in any domain can communicate with another host in another domain. This basically forms the hierarchical structure of Internet. However, at times during its development the Internet topology appeared to have some exceptions. Some stub domains may want to build lateral links because of easy exchange of information. For example, a lateral link between two universities is for research purpose or between two companies is for cooperated business. The lateral links could occur at any level of hierarchy (e.g., direct links between WAN's, between LAN's, and between stubs). Another exception also exists when some stub domains not only have a link to a LAN, but also have a bypass link to a WAN. Figure 5.1 shows the structure of the current Internet.

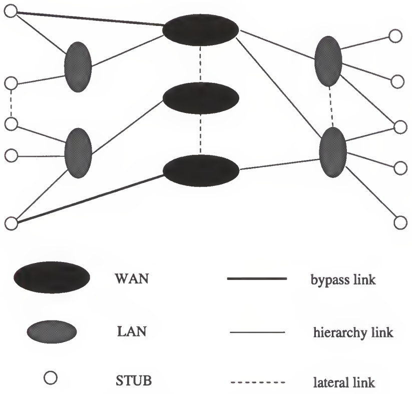


Figure 5.1. The Internet topology

5.2 Internet Topology and Directed Graph Mapping

This Section tries to map the Internet topology to a DG defined in Chapter 4. Each domain in Figure 5.1 is mapped as a node (class) in the DG and each link in Figure 5.1 is mapped as one or two positive edges. It is a straightforward one-to-one mapping between domains and nodes. The mapping between links and positive edges needs to consider the following two cases.

1. Either a hierarchy or a bypass link is mapped as a positive edge with arrow pointing to the lower level node.
2. A lateral link is mapped as two positive edges pointing to different directions.

The first case means higher level domains can derive lower level domains' secret keys. Since a lower level domain requires a higher level domain's transit service and packets originating from the lower level domain are usually protected by its secret key. Therefore, it is necessary to grant higher level domains the privilege to derive lower level domains' secret keys. The second case occurs at the same level and two positive edges means that both domains can derive the other's secret key. This assumption is reasonable because a lateral link connects two domains requiring each domain to provide transit service (for transit domains) or process data (for stub domains) for the other. The mapping discussed so far only includes the positive edges, Figure 5.2 shows the DG mapping from the Internet topology in Figure 5.1. In the lateral link mapping, we use one double-arrow edge instead of two positive edges for clarity.

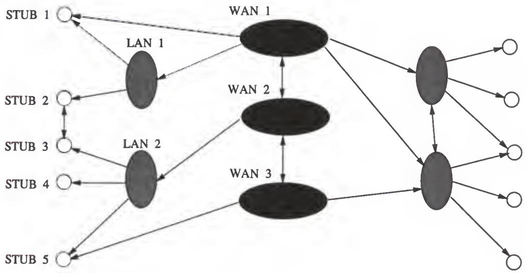


Figure 5.2. The DG with only positive edges

The mapping from a lateral link to a double-arrow edge forms access cycle between two domains. In the definition of DG in Chapter 4, these two domains should be treated as the same class. This is bad because these two domains are actually under different administrative authority. Fortunately, some negative edges can be added to make them different, though they can access each other.

In figure 5.2, WAN 1 can derive all secret keys of its subtree of the hierarchy (i,e., LAN 1, STUB 1, and STUB 2). Likewise, WAN 2 can derive the secret keys of LAN 2, STUB 3, STUB 4, and STUB 5. If there is a double-arrow edge between WAN 1 and WAN 2, this means that both WAN1 and WAN 2 can derive the secret keys of the other's subtree. If WAN 1 (or WAN 2) is compromised, all communications among domains within WAN 2's (or WAN 1's) subtree could be compromised. This is not desirable because it violates the "need to know" (a form of least privilege) security principle. Therefore, it is better not to give a domain the access privilege beyond the lateral link. Negative edges can be used to break the transitive property of the double arrow edges so that a domain can not access beyond the lateral links.

In Chapter 4, a negative edge connecting two classes only indicates that the source class can not derive the destination class' key. It is a "one-to-one" relationship between classes, i,e., a specific source class to a specific destination class. In the application of internetwork domain discussed above, a class may not have the privilege to access a set of classes. For example, WAN 2 should not access LAN 1, STUB 1, and STUB 2. If we use the negative edge defined in Chapter 4, three negative edges, which point from WAN 2 to LAN 1, STUB 1 and STUB 2 respectively, are necessary to represent the transitive exceptions. For the DG representation clarity of Internet domain access control policy, we redefine the negative edge to be an edge pointing from a source node to a virtual destination node. The virtual destination node represents the set of nodes that should not be accessed by the source node. Figure 5.3 shows the DG with negative edges pointing to virtual nodes using the same Internet topology example.

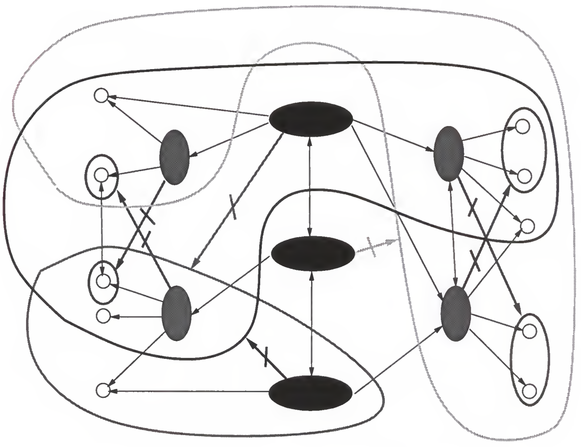


Figure 5.3. The DG with negative edges and virtual nodes

This kind of DG greatly reduces the number of negative edges needed to represent the access policy of the internetwork domain application. In the example above, only 7 negative edges are needed, though there are 32 transitive exceptions.

Having an Internet domain access control policy like the one shown in Figure 5.3, a new key generation scheme can be used to assign keys to each domain for secure session key distribution. The scheme to distribute the session key is addressed in next section.

5.3 Safe Session Key Distribution

After the secret keys are assigned to each domain, the session key for any communication session can be safely distributed. We use an example in Figure 5.2 to demonstrate the distribution of a session key. A source host in STUB 2 wants to communicate with a destination host in STUB 5. The path setup initiates from STUB 2, and goes through LAN 1, WAN 1, WAN 2, WAN 3, then STUB 5. If all stub and transit domains allow communication between the source and destination hosts, STUB 5 generates a session key and distributes it to all domains in the path. Figure 5.4 illustrates the scenario of session key distribution.

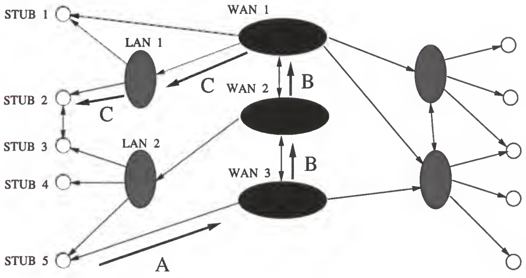


Figure 5.4. Packet A contains $< K_s >_{K^e_{stub\ 5}}$ which means that the session key K_s is encrypted by the assigned encryption key $K^e_{stub\ 5}$ of stub 5; and Packet B and C contain $< K_s >_{K^e_{wan\ 3}}$ and $< K_s >_{K^e_{stub\ 2}}$ respectively

First, STUB 5 uses its assigned encryption key to encrypt the session key and forwards it to WAN 3 in packet *A*. WAN 3 is capable of deriving STUB 4's encryption key and decrypting the session key. After getting the session key, WAN 3 should use its own encryption key to re-encrypt the session key in packet B and forward it to WAN 2. Upon receiving packet B from WAN 3, WAN 2 derives WAN 3's encryption key to decrypt the session key. At this point, there are two possible access control assumptions:

1. WAN 1 is able to derive WAN 3's encryption key. A communication path could contain a long list of consecutive WANs. If WANs cannot access across lateral links among WANs, the session key re-encryption occurs at each WAN and slows down the communication.
2. WAN 1 is not able to derive WAN 3's encryption key. This assumption sacrifices performance to achieve better security.

In Figure 5.4, we use the first assumption and WAN 2 just forwards packet B to WAN 1. There is no re-encryption necessary for WAN 2. In the first assumption, WAN 1 is able to derive WAN 3's encryption key and decrypt the session key. Instead of using its own encryption key, WAN 1 derives STUB 1's encryption key to encrypt the session key in packet C and forwards the packet to LAN 1. LAN 1 derives STUB 1's encryption key to decrypt the session key and forwards packet C to STUB 1. Finally, STUB 1 uses its own encryption key to decrypt the session key. Of course, the packet header should contain a node ID field indicating who encrypts the session key so that the receiving nodes is able to derive the corresponding encryption key.

We assume no preference for either assumption, though the example above uses the first one for demonstration. Which assumption should be used depends on whether security or performance is more important to the Internet communication.

In Chapter 2, we list two kinds of IAC protocols - KIAC and TIAC. Example above is based on the KIAC protocol. For the TIAC protocol, the key distribution process is similar except that each domain gateway puts the session key into the ticket instead of storing it in local memory. The number of session key encryptions in KIAC is equal to the number of ticket layers in TIAC. In the example of Figure 5.4, the session key encryptions occurred at STUB 5, WAN 3 and WAN 1. Thus, three layers are necessary in the ticket for the TIAC protocol.

5.4 Chapter Summary

In this chapter, we demonstrate how to use the key assignment scheme for safe session key distribution, as well as the mapping from Internet topology to interdomain access DG. The assumption of access privileges for an internetwork domain in Section 5.2 requires further detailed discussion such as “does the access transitive property stop on all lateral links?”, “what kind of bypass links can be ignored in the mapping?”, and so forth.

The immediate work of this research is to categorize all reasonable assumptions comprehensively. In the next Chapter, long term future work about interdomain access control is discussed.

CHAPTER 6
SUMMARY AND FUTURE WORK

This dissertation examines security and efficiency problems of interdomain access control (IAC) across heterogeneous administrative domains (ADs). With different administrative policy for each domain, only authenticated and authorized packets are allowed to flow through domain boundaries.

6.1 Summary

My first contribution for interdomain access control is to develop a key-based and a ticket-based IAC protocols, which integrate the underlying policy routing facility IDPR (InterDomain Policy Routing). These two protocols build a communication path from source to destination and solve packet-dropping problems occurred in intermediate nodes.

For communication path setup efficiency, an active network infrastructure is better than current TCP/IP. My second contribution is to develop a dynamic path setup protocol in an active network environment to reduce memory overhead.

Every data packet flow through a path relies on a session key for authentication and authorization. Therefore, the secrecy of the session key is essential for interdomain access control. My third contribution is to develop a generic key assignment scheme for enforcing policy exceptions that can be utilized to assign keys to administrative domains so that the session key can be safely distributed.

6.2 Future Work

This dissertation proposed several IAC protocols utilizing different combinations of static/dynamic and stateful/stateless approaches under the current TCP/IP and the future active network infrastructures. Active networks are a novel network architecture in which each switch in the network provides a computation environment such that customized computation can be performed on the fly based on the messages flowing through them. In essence, the network becomes programmable for any specific application. It is a revolutional concept that assumes a store-compute-forward networking paradigm.

Active networks are still in developing stage and not mature enough. In order to deploying active networks, some open issues need to be discussed comprehensively and designed carefully such as secure resource management, active program encoding, efficient active node architecture, active node placement, and so forth. My long term future work is to study the security issues of diverse applications under the active network prototype. Hopefully, some results of the study can benefit the development and deployment of the active network infrastructure itself.

Some short term future work related to interdomain access control are listed in the following sections.

6.2.1 Neighbor Selection Criteria in Dynamic Path Setup

The dynamic path setup protocol described in Chapter 3 does not address the neighbor selection criteria. To repair a failed communication session, a fast path repair process is crucial. The process should be efficient in failure detection and recovery. The proposed protocol in Chapter 3 achieves fast detection. However, failure recovery requires a fast detour path setup and relies on a good QNL selection criteria. Each router in the path setup protocol has no knowledge of the QNL in the selected next router. If the QNL is empty, a Negative Message may be returned and cause a rewinding of the search. This kind of path setup rewinding should be limited by a good selection criteria. One way to decrease the possibility of the path setup rewinding is to increase the information available to each router. For example, the past history of previous path setup or QNL of the neighbors. To make this information available to each router may slow down the path setup in another respect. Therefore, future work of this protocol is to find a good QNL selection criteria.

6.2.2 Multiple Failures in Dynamic Path Setup

Multiple failures is another issue not addressed in the protocol. There may be multiple routers detecting different failures simultaneously. It is not a good idea to have multiple path repair processes running at the same time. Simultaneous repairs may result in redundant work and even incorrect path repair due to interference. This is also an open area to be addressed.

6.2.3 Administrative Policy Encoding and Fast Retrieval

In previous chapters, we have described three IAC protocols. All of them are trying to find a path so that all the involved gateways (routers) can satisfy the source request service. The process of the path setup actually is performing the administrative policy match-up among source, intermediate, and destination domains. The questions are what is the administrative policy and how the policy match-up is performed.

Clark [5] suggests that each policy is described in English and then expressed in a Policy Term (PT) notation. Each PT encodes a distinct policy of an Administrative Domain (AD) that synthesized it. The form of a PT is:

$$[(H_{end}, AD_{end}, AD_{ent}), (H_{end}, AD_{end}, AD_{exit}), UCI, Cg]$$

where:

H_{end} is the source or destination end system,

AD_{end} is the source or destination AD,

AD_{ent} is the traffic entering AD,

AD_{exit} is the traffic exit AD,

UCI is the user class identifier,

C_g are any global conditions.

Each PT specifies that the communication is allowed between two specific end hosts in two domains if the communication traffic is entering from AD_{ent} and exit through AD_{exit}.

The UCI makes PT's more fine-grained from host-based to user-based and C_g concerns all global conditions such as billing or quality of service (QoS). Note that, both UCI's and C_g's should be globally defined.

The exponential growth of the Internet has brought diverse users, ADs, and applications. Therefore, the PT should be powerful and flexible enough to express the complex policies in the current highly dynamic Internet environment. The PT defined by Clark, above, is a basic form and definitely not enough for the future.

Not only is the PT encoding itself important, but also the data structure of PTs within a gateway (router) needs further study so that the retrieval can be fast.

GLOSSARY

ACM	Access Control Matrix
ACP	Access Control Policy
AD	Administrative Domains
AS_i	Accessible Set of a class C_i
CAP	Capsule
DG	Directed Graph
COM	Computational Complexity: number of encryptions and decryptions
DS_i	Dominating Set of a class C_i
E(X,K)	Plaintext X encrypted by key K
FID	Forwarding Information Database
H	Host
IAC	Interdomain Access Control
IDPR	InterDomain Policy Routing
K_i	Secret key known only to PG_i
K_{ij}	Secret key shared by PG_i and PG_j
K_s	Secret session key for a traffic flow
KIAC	Key-based Interdomain Access Control
LAN	Local Area Network
MAC	Message Authentication Code
MAC(K)	packet's Message Authentication Code is a signed message digest of the packet's contents using key K

MAN	Metropolitan Area Network
MTU	Maximum Transfer Unit
n-system	A system with n distinct user classes
NT	Nodes Traversed
NW	Network Load: number of packets $\times$ number of PGs traversed
PFP	Packet Forwarding Protocol
PG	Policy Gateway
poset	Partially Ordered Set
PR	Policy Route
PS	Path Status
PSP	Path Setup Protocol
PT	Policy Term
PUP	Policy Update Protocol
QNL	Qualified Neighbor List
QoS	Quality of Services
RID	Routing Information Database
RSA	A public key cryptographic algorithm developed by R. L. Rivest, A. Shamir and L. Adleman
RT	Router
SA	Security Association
SEC	Secret Key Searching: number of secret key database searches
SES	Session Key Searching: number of session key database searches
SKD	Session Key Database

SRS	Source Requested Service
T^i_{ab}	(n-i+1)th partial ticket generated by PG_i for traffic from H_a to H_b
T^i_{ba}	i-th partial ticket generated by PG_i for traffic from H_b to H_a
TIAC	Ticket-based Interdomain Access Control
UCI	User Class Identifier
UM	User Matrix: A matrix represents the access control policy
WAN	Wide Area Network

REFERENCES

[1] S. G. Akl and P. D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, August 1983.

[2] C. Alaettinoglu and A. U. Shankar. The viewserver hierarchy for interdomain routing: Protocols and evaluation. *IEEE Journal on Selected Areas in Communications*, 13(8):1396–1410, October 1995.

[3] S. Bhattacharjee, K. Calvert, and E. Zegura. An architecture for active networking. In *Technical Report GIT-CC-96-20*, College of Computing, Georgia Institute of Technology, 1996.

[4] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security.* Addison-Wesley, 1994.

[5] D. Clark. Policy routing in internet protocols. RFC 1102, May 1989.

[6] D. Estrin, M. Steenstrup, and G. Tsudik. A protocol for route establishment and packet forwarding across multidomain internets. *IEEE/ACM Transactions on Networking*, 1(1):56–70, February 1993.

[7] D. Estrin and G. Tsudik. VISA scheme for inter-organization network security. In *Proceedings of the 1987 Symposium on Security and Privacy*, pages 174–183, 1987.

[8] L. Harn, Y.-R. Chien, and T. Kiesler. An extended cryptographic key generation scheme for multilevel data security. In *Fifth Annual Computer Security Application Conference*, pages 254–262, 1989.

[9] L. Harn and H. Y. Lin. A cryptographic key generation scheme for multilevel data security. *Computers and Security*, 9(6):539–546, 1990.

[10] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Communications ACM*, 19(8):461–471, August 1976.

[11] M.-S. Hwang and W.-P. Yang. A new dynamic cryptographic key generation scheme for a hierarchy. In *1994 IEEE Region 10's Ninth Annual International Conference*, pages 465–468.

[12] M. S. Iqbal and F. S. F. Poon. Packet level access control scheme for internetwork security. *IEE Proceedings-1*, 139(2):165–175, April 1992.

[13] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World.* Prentice Hall, 1995.

[14] H.-T. Liaw. A dynamic cryptographic key generation and information broadcasting scheme in information systems. *Computers and Security*, 13(7):601–610, 1994.

[15] S. J. Mackinnon, P. D. Taylor, H. Meijer, and S. G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transactions on Computers*, c-34(9):797–802, September 1985.

[16] H. Park and R. Chow. Internetwork access control using public key certificates. In *Proceedings of IFIP SEC 96 12th International Security Conference*, pages 237–246, May 1996.

[17] M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. In *Technical Report MIT/LCS/TR-212*, Laboratory for Computer science, Massachusetts Institute of Technology, Cambridge, Mass., January 1979.

[18] Y. Rekhter. Inter-domain routing protocol. *J. Internetworking Res. Experience*, 4:61–80, 1993.

[19] Y. Rekhter and T. Li. A border gateway protocol 4. RFC 1654, July 1994.

[20] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications ACM*, 21:120–126, February 1978.

[21] E. C. Rosen. Exterior gateway protocol. RFC 827, October 1982.

[22] R. S. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27:95–98, 1988.

[23] B.-M. Shao, J.-J. Hwang, and P. Wang. Distributed assignment of cryptographic keys for access control in a hierarchy. *Computers and Security*, 13(1):79–84, February 1994.

[24] M. Steenstrup. Inter-domain policy routing protocol specification: Version 1. RFC 1479, July 1993.

[25] D. L. Tennenhouse, S. J. Garland, L. Shrira, and M. F. Kaashoek. From internet to activenet. Request for Comments, January 1996.

[26] D. L. Tennenhouse and D. J. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26(2), April 1996.

[27] H.-M. Tsai and C.-C. Chang. A cryptographic implementation for dynamic access control in a user hierarchy. *Computers and Security*, 14(2):159–166, 1995.

[28] Y.-M. Tseng and J.-K. Jan. A scheme for authorization inheritance in a user hierarchy. In *1997 Information Security Conference (INFOSEC '97)*, pages 109–115.

[29] P. F. Tsuchiya. The landmark hierarchy: A new hierarchy for routing in very large network. In *Proceedings ACM SIGCOMM '88*, August 1988.

[30] F.-K. Tu, C.-S. Laih, and W.-C. Kuo. Cryptanalysis of a new cryptographic solution for dynamic access control in a hierarchy. In *1997 Information Security Conference (INFOSEC '97)*, pages 104–108.

[31] S.-J. Wang and J.-F. Chang. A hierarchical and dynamic group-oriented cryptographic scheme. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E79-A(1):76–85, January 1996.

[32] D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. Ants: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, April 1998.

[33] T.-C. Wu, T.-S. Wu, and W.-H. He. Dynamic access control scheme based on the Chinese remainder theorem. *Computer Systems Science and Engineering*, 10(2):92–99, April 1995.

[34] J.-H. Yeh, R. Chow, and R. Newman. Interdomain access control with policy routing. In *Proceedings of Sixth IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pages 46–52, October 1997.

BIOGRAPHICAL SKETCH

Jyh-haw Yeh was born on May 20, 1966, in Taoyuan, Taiwan, Republic of China. He received his Bachelor of Science degree from National Chung-Ching University, Taichung, Taiwan, Republic of China, in 1988. He received his Master of Science degree from Cleveland State University, Cleveland, Ohio, in 1993. He received his Doctor of Philosophy degree in Computer and Information Science and Engineering at the University of Florida, Gainesville, Florida, in December 1999. His research area is computer systems with specialties in network security, network access control, and active networks.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Randy Y. C. Chow, Chairman
Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Richard Newman
Assistant Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Timothy A. Davis
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Eric Hanson
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Mark C. K. Yang
Professor of Statistics

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

December, 1999

M. J. Ohanian
Dean, College of Engineering

Winfred M. Phillips
Dean, Graduate School